STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

PROSESSINHALLINTAJÄRJESTELMÄ MS ACCESSILLA JA VBA:LLA

Työn tekijä: Tomi Sarpola
Työn valvoja: Juhani Rajamäki
Työn ohjaaja: Sari Tervo

Työ hyväksytty: __. __. 2007

Juhani Rajamäki
Lehtori

STADIA
HELSINGIN AMMATTIKORKEAKOULU

INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä:
Tomi Sarpola

Työn nimi:
Prosessinhallintajärjestelmä MS Accessilla ja VBA:lla

Päivämäärä: 23.11.2007 Sivumäärä: 56

Koulutusohjelma: Tieto- ja sähkötekniikka
Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn valvoja: Juhani Rajamäki, lehtori, Helsingin Ammattikorkeakoulu Stadia

Työn ohjaaja: Sari Tervo. prosessiomistaja, Nokia Siemens Networks

Tässä insinöörityössä suunniteltiin ja toteutettiin Nokia Oyj:n Networks -toimialaryhmään kuuluvalle toimitusvalmiusprosessin kehitysryhmälle prosessinhallintaohjelmisto.

Ohjelmiston tarkoituksena oli yksinkertaistaa prosessiin kuuluvan sähköisen materiaalin päivitystyötä, vähentää prosessipäivityksiin tarvittavaa työmäärää sekä parantaa sähköisen materiaalin käyttöä tuoteprojekteissa. Insinöörityö toteutettiin ohjelmistoprojektina, johon sisältyi tutkimusvaihe, jossa tutkittiin prosessinkehitykseen liittyvää yritystoimintaa ja eri teknologiavaihtoehtoja tuotettavan järjestelmän toteuttamiselle.

Työssä esitellään tulokset tutkimuksesta, jossa ensin tutustuttiin prosessikehityksen työkaluihin tilaajan tietojärjestelmissä ja prosessikehitykseen liittyvästä yritystoiminnasta. Toteutusteknologian tutkimuksessa perehdyttiin Microsoft Office -tuoteperheeseen kuuluvien MS Excelin ja MS Accessin sovelluksenkehityksen mahdollisuuksiin. Tutkimuksessa keskityttiin Visual Basic for Applications -ohjelmointiin, sovellusten hajauttamiseen tiedostopalvelinympäristössä sekä Visual Basic for Applications-ohjelmoinnin uudelleenkäytettävyyteen ja ohjelmiston hallittavuuteen. MS Office -tuotteiden vaihtoehtoina tutkittiin itsenäisten Visual Basic-sovellusten, C++- ja Java-sovellusten käyttöä sekä MySQL-tietokantajärjestelmän toimintoja.

Työssä käydään myös läpi projektissa tuotettu Prosessinhallintajärjestelmä, joka on MS Accessilla toteutettu VBA-kielinen tietokantasovellus. Järjestelmä toteutettiin tutkimusten perusteella tiedostopalvelinympäristöön soveltuvaksi hajautetuksi tietokantasovellukseksi, johon kuului keskitetyn tietokannan lisäksi hallintasovellus, asiakassovellus ja siihen liittyvä asiakkaan tietokanta.

Prosessinhallintajärjestelmä valmistui keväällä 2007 ja otettiin käyttöön Nokia Siemens Networksillä syksyllä 2007.

Avainsanat: prosessi, prosessinmallinnus, Nokia, Nokia Siemens Networks, tietokanta, Visual Basic for Applications, VBA, Microsoft Office, MS Access, MS Excel, QPR, Process Guide, Process Portal, järjestelmän hajautus, uudelleenkäytettävyys, MVC

STADIA
HELSINGIN AMMATTIKORKEAKOULU

ABSTRACT

Name:
Tomi Sarpola

Title:
Process Management Applications with MS Access and VBA

Date: 23.11.2007 Number of pages: 56

Department: Department of Technology
Study Programme: Software Engineering

Instructor: Juhani Rajamäki, Lecturer, Helsinki Polytechnic University Stadia

Supervisor: Sari Tervo, Process Owner, Nokia Siemens Networks

The objective in this graduate study was to design and implement a process management Applications for the Delivery Capability Process development team of Nokia Oyj.

The purpose of the Applications was to simplify the work needed for handling of process related material, to reduce the work amount needed for updating the process content to process related material, and to improve the use of process material in product programs. The graduate study was executed as a software development project. The project included a study phase for collecting information about the business processes related to process development and the technology alternatives for implementing the process management Applications.

The graduate study presents the results of the preliminary study, which consisted of an introduction to the process development tools available at the Nokia IT-environment and the business processes related to process development. The study for development technologies analyzes the Applications development features of MS Excel and MS Access, which are part of the Microsoft Office product family. The study concentrates on Visual Basic for Applications programming, distributed data processing Applications and VBA Applications re-usability and manageability. As alternatives for the MS Office products, the study also includes an introduction to stand-alone Visual Basic and VB.NET Applications, to C++- and Java database Applications and to MySQL database management system.

This study also includes a presentation of the Process Management Applications that is a database application developed with VBA on MS Access. The Application was developed based on the study as a distributed database system that operates in a fileserver environment. The system consists of centralized database, process administration Applications, client Applications for product programs and client database.

The Process Management Applications was finished during the spring 2007 and was taken into use at Nokia Siemens Networks during the fall 2007.

Keywords: process, process modeling, Nokia, Nokia Siemens Networks, database, Visual Basic for Applications, VBA, Microsoft Office, MS Access, MS Excel, QPR, Process Guide, Process Portal, system distribution, re-usability, MVC

LYHENNE- JA MÄÄRITELMÄLUETTELO

Ad hoc	*Ad hoc* on latinaa ja tarkoittaa käännettynä "tätä (tarkoitusta, tehtävää) varten". Sanonta on yleisesti tietotekniikassa käytetty termi, jolla kuvataan tiettyä ongelmaa varten räätälöityä ohjelmistoratkaisua tai sen toimintoja.
ADO	*Microsoft ActiveX Data Objects*, joukko tietokantayhteyden mahdollistavia COM-objekteja jotka toimivat rajapintana sovelluksen ja OLE DB:n välillä.
API	*Applications programming interface*, Ohjelmointirajapinta on liittymä jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja keskenään.
COM	*Component Object Model*, Microsoftin kehittämä ohjelmistokomponenttien alusta joka mahdollistaa prosessien välisen kommunikoinnin sekä dynaamisen objektien luonnin millä tahansa teknologiaa tukevalla ohjelmointikielellä.
DAO	*Microsoft Data Access Objects library*, virallisesti käytöstä poistunut Microsoftin kehittämä tietokantayhteyden mahdollistava oliopohjainen ohjelmointirajapinta joka toimii sovelluksen ja JET DB engine -alustan välillä.
DCP	*Delivery Capability Process*, tuoteprosessiin kuuluva toimitusvalmiuteen tähtäävä prosessi.
DLL	*Dynamic Linking Library*, on Microsoft Windowsin käyttämä jaetun kirjaston tyyppi. Jaetuilla kirjastoilla käyttöjärjestelmä jakaa ohjelmakoodia ja dataa eri ohjelmien kesken.
GPL	*GNU General Public Lisence*, GNU yleinen lisenssi on vapaa ohjelmistolisenssi, jonka tarkoitus on taata käyttäjälle oikeus kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia
IDE	*Integrated Development Environment*, Integroitu ohjelmointiympäristö on ohjelmistotuotannon mahdollistava ohjelma tai joukko ohjelmia, ja se sisältää yleensä toiminnot lähdekoodin muokkaukseen, kääntäjän tai tulkin, ohjelmistojakelun kokoamiseen sekä useimmiten myös virheiden poistoon ja etsintään.
JAVA	Java on Sun Microsystemsin kehittämä laaja teknologiaperhe ja ohjelmistoalusta, johon kuuluu mm. laitteistoriippumaton oliopohjainen ohjelmointikieli, ajoaikainen ympäristö virtuaalikoneineen ja luokkakirjastoineen.
JET	*Microsoft Joint Engine Technology Database Engine*, on Microsoftin tietokantoja tukeva tietokantamoottori, se vastaa alemman tason

	tietokantatransaktioista relaatiotietokantojen hallintajärjestelmissä, ja tarjoaa muille ohjelmille rajapinnan relaatiotietokantojen manipulointiin.
LNC	*Leszynski naming convention,* on MS Accessissa yleisin nimeämiskäytäntö jossa käytetään lyhyitä etuliitteitä muuttujien määrittelyyn.
LOB SW	*Line-of-business software*, sovellukset tai sovellusten osat jotka tuottavat yrityksen johdolle kriittisiä palveluita ja hallintatyökaluja, kuten toimitusketjun ylläpidon- ja resurssienhallinnan sovellukset
ODBC	*Open Database Connectivity*, standardi ohjelmointirajapinta relaatiotietokannan tietoyhteyksiä varten
RAD	*Rapid Applications Development*, tehokas ja nopea iteratiivista mallia sekä prototyyppien tuotantoa hyödyntävä ohjelmistokehitysprosessi.
RDBMS	*Relational Database Management System*, on relaatiotietokannan hallintaan tarkoitettu sovellusohjelma, joka hallinnoi tiedon organisoinnin, hallinnan, varastoinnin ja esityksen transaktioita. RDBMS järjestelmät tukevat relaatioiden määrittelyä ja tallennusta sekä relaatiota käyttävää tietokantakyselykieltä kuten SQL:ää.
SQL	*Structured Query Language*, standardoitu kyselykieli relaatiotietokannoille.
UML	*Unified Modeling Language*, Mallinnusmenetelmä, jolla voidaan suunnitella ja kuvata ohjelmistojen sekä tietokantojen rakennetta.
VB	*Visual Basic*, on suosittu Microsoftin kehittämä tapahtumakeskeinen, helppokäyttöisyyteen ja nopeaan sovelluskehitykseen (RAD) tähtäävä BASIC -sukuinen ohjelmointikieli.
VBA	*Visual Basic for Applications*, on VB:n implementaatio jota käytetään pääasiassa MS Office ohjelmistoihin ja AutoCAD:iin integroidun kehitysympäristön (IDE) kautta OLE Automation-rajapinnan ohjelmointiin, joka mahdollistaa isäntäsovelluksen toimintojen sekä käyttöliittymän hallinnan.
VBE	*Visual Basic Editor*, on MS Officeen integroitu VBA sovellusten kehitysympäristö (IDE) Visual Basic Editor-ohjelman, joka otettiin mukaan MS Office 97 -versiossa.

SISÄLLYS

1 JOHDANTO

Tämän insinöörityön tavoitteina oli tutkia eri teknologiavaihtoehtoja ja menetelmiä prosessin hallintaan soveltuvan kustannustehokkaan järjestelmän toteutukseen sekä suunnitella ja toteuttaa ohjelmisto prosessin hallintaan tutkimuksesta saatujen tulosten perusteella.

Työn tilaajana toimi Nokia Oyj:n Networks -toimialaryhmään kuuluva toimitusvalmiusprosessin kehitysryhmä, jolle työn tekeminen aloitettiin syksyllä 2006. Vuoden 2007 huhtikuussa Networks -toimialaryhmä siirtyi osaksi uutta verkkoalan yritystä, Nokia Siemens Networks Oy:tä.

Toteutettavan prosessinhallintaohjelmiston tarkoituksena on yksinkertaistaa prosessiin kuuluvan sähköisen materiaalin päivitystyötä, vähentää prosessipäivityksiin tarvittavaa työmäärää sekä parantaa sähköisen materiaalin käyttöä Nokia Networksin tuoteprojekteissa.

Työssä tutustutaan ensin prosessin kehityksen työkaluihin tilaajan tietojärjestelmissä sekä prosessikehitykseen liittyvään yritystoimintaan. Sen jälkeen tutkitaan Microsoft Office -tuotteiden, MS Excelin ja MS Accessin ominaisuuksia, Visual Basic for Applications -ohjelmointia, sovellusten hajauttamista tiedostopalvelinympäristössä sekä perehdytään ohjelmakoodin uudelleenkäytettävyyteen ja ohjelmiston hallittavuuteen VBA-ohjelmoinnin menetelmiä käyttäen. MS Office -tuotteiden vaihtoehtoina tutkittiin itsenäisten Visual Basic-sovellusten, C++- ja Java-sovellusten käyttöä sekä MySQL-tietokantajärjestelmän toimintoja.

2 NOKIA NETWORKS JA TOIMITUSVALMIUSPROSESSI

Työn tilaajana toimi Nokia Oyj:n osasto, joka vastaa Nokia Networks -toimialaryhmän tuoteprosessiin kuuluvan toimitusvalmiuteen (DCP) tähtäävän prosessin kehityksestä.

2.1 Yritysfuusio

Työ aloitettiin Nokia Oyj:lle syyskuussa 2006. Työn suorituksen aikana keväällä 2007 Nokia Oyj ja Siemens AG yhdistivät tietoliikenneverkoissa käytettävien laitteiden suunnittelusta ja valmistuksesta vastaavat jaoksensa: Nokian Networks -toimialaryhmä ja Siemensin COM-divisioona lukuun ottamatta sen Enterprise Business -yksikköä. Yhteisen verkkoyhtiön Nokia Siemens Networks Oy:n toiminta oli tarkoitus aloittaa joulukuun loppuun 2006 mennessä. Virallisesti yritys aloitti toimintansa 1.4.2007.

Insinöörityön toteutuksen kannalta yhdistyminen ilmeni poikkeavien ja muuttumassa olevien käytäntöjen sekä tiukentuneiden turvatoimien muodossa, jotka osaltaan vaikuttivat projektin määrittelyyn, resursointiin ja päivittäiseen asiointiin. Kattavien IT-palveluiden ansiosta projektista ja työn seurannasta saatiin joustavaa ja projektissa pystyttiin keskittymään olennaiseen.

2.2 Toimitusvalmiusprosessin kehitysryhmä

Toimitusvalmiusprosessin kehitysryhmä vastaa Nokia Networksin toimitusvalmiusprosessin suunnittelusta, dokumentoinnista ja julkaisuista. Ryhmä toimii tuoteprosessin kehitysryhmän rinnalla, joka taas vastaa koko Nokia Networksin tuoteprosessin kehitysprojektien aikatauluista, aliprosessien hyväksynnästä sekä prosessinkehitystyökaluista.

Toimitusvalmiusprosessin kehitysryhmä tilasi insinöörityön, koska nykyiseen kehitysympäristöön tarvittiin prosessinhallintaa helpottava ratkaisu, joka vähentäisi prosessipäivityksiin tarvittavaa työmäärää.

2.3 Prosessinhallinnan kehitystavoitteet

Prosessin päivityksiin liittyi runsaasti tiedonkäsittelyä, joka olemassa olevilla työkaluilla oli hankalaa toteuttaa. Tilatun insinöörityön tavoitteina oli tutkia eri teknologiavaihtoehtoja ja menetelmiä prosessin hallintaan soveltuvan kustannustehokkaan tietojärjestelmän toteutukseen sekä suunnitella ja toteuttaa

prosessinhallintatyökalu, jonka tarkoituksena on yksinkertaistaa prosessiin kuuluvan sähköisen materiaalin päivitystyötä sekä parantaa sähköisen materiaalin käyttöä tuoteprojekteissa.

Tutkimuksen pohjana käytettiin aiemmin toteutettujen järjestelmien tarjoamia ominaisuuksia. Toteutusten ja lähdemateriaalin perusteella tutkittiin MS Office - ja VBA-ohjelmien soveltuvuutta prosessinhallintaan. Yritystoiminnan tutkimuksessa perehdyttiin Nokia Networksin prosessinkehityksen työkaluihin ja kehitysympäristöön sekä niiden asettamiin vaatimuksiin.

Insinöörityöprojekti sisälsi tutkimusvaiheen lisäksi tyypillisen ohjelmistoprojektin vaiheet mukaan lukien käyttöönoton organisoinnin ja pääkäyttäjien koulutuksen. Nokia Oyj:n ja Nokia Siemens Networks Oy:n yrityskielenä on englanti, projektissa toteutettua sovellusta käsittelevä osuus tuotettiin ja dokumentoitiin englanniksi.

2.4 Toimitusvalmiusprosessin sähköinen materiaali

Toimitusvalmiusprosessin sähköinen materiaali koostuu prosessin toimintojen kuvauksista ja prosessin suoritukselle asetetuista vaatimuksista. Materiaalia käsitellään useassa eri ympäristössä.

Tuotteenvalmistusprosessin kehitysryhmän määrittelemässä ympäristössä käsitellään koko tuotteenvalmistusprosessia koskevat materiaalit ja niiden versioiden hallinta:

- prosessiaktiviteettien toimintakuvaukset
- etappivaatimusten listaukset.

Toimitusvalmiusprosessin kehitysryhmän ylläpitää itsenäisesti toimitusvalmiusprosessin käyttöönottoon ja koulutukseen liittyviä materiaaleja:

- kolmelle eri tuoteprojektityypille (HW, SW, Iterative SW) tuotetut prosessin käyttöoppaat, joita käytetään prosessin suorituksen ohella
- prosessin koulutusmateriaali
- raporttipohjat prosessin seurantaan.

Tuoteprojektikohtaiset käyttöoppaat sisälsivät projektityypille olennaiset aktiviteettikuvaukset sekä etappivaatimukset toisiinsa linkitettynä. Oppaiden tarkoituksena on auttaa aktiviteettikuvausten ja etappivaatimusten käyttöönot-

toa tuoteprojekteissa tarjoamalla suoraviivaisen mallin toimitusvalmiusprosessin määrittelyyn ja aikatauluttamiseen. [11.]

2.5 Prosessin mallinnusympäristö ja projektin lähtökohta

Nokia Networksin tuotteenvalmistusprosessin kehitysryhmä määrittelee työkalut, joita tuotteenvalmistusprosessin ja sen aliprosessien kehitykseen käytetään [11].

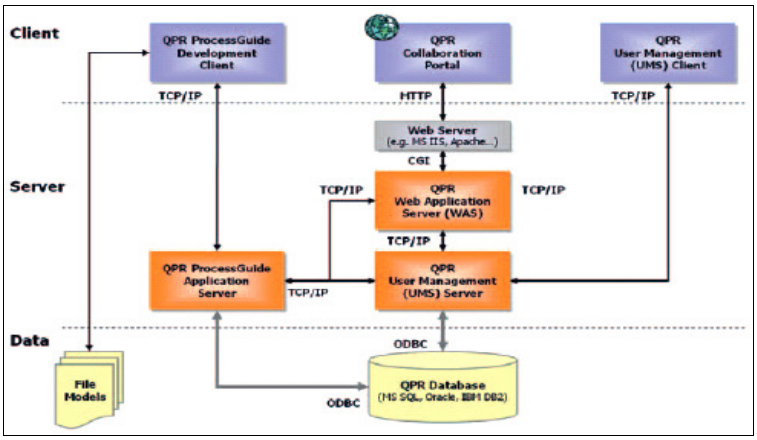
2.5.1 Prosessin etappivaatimusten määritys MS Excelissä

Prosessin etappivaatimusten määrittely tapahtuu MS Excelissä. Vaatimukset listataan aliprosesseittain ja järjestetään etappiaikataulun perusteella ryhmiin.

Kunkin aliprosessin kehitysryhmä määrittää omistamansa prosessin vaatimukset, jotka ensin katselmoidaan sisäisesti ja lopulta esitetään tuoteprosessin kehitysryhmälle hyväksyttäväksi. Tuoteprosessin kehitysryhmä kerää ja luetteloi vaatimukset tai niihin tehdyt muutokset julkaistavaan listoihin, jotka sisältävät koko tuotteenvalmistusprosessin etappivaatimukset yhdestä tuotteenvalmistusvaiheesta. [11].

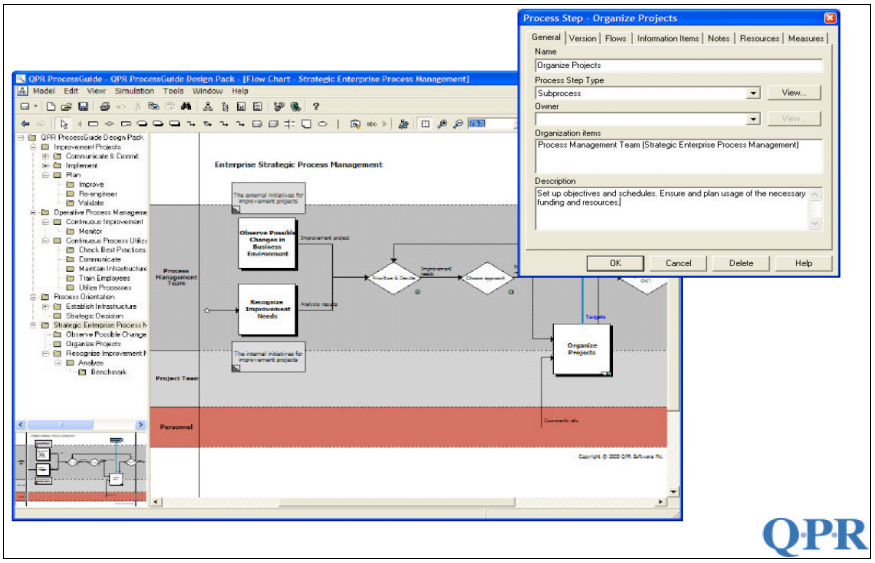
2.5.2 Prosessiaktiviteettien kuvaus QPR Process Guide sovelluksessa

Prosessin aktiviteetit kuvataan QPR Process Guide -ohjelmistossa. Process Guide on QPR Software Plc:n kehittämä monipuolinen ja visuaalisesti edistynyt prosessinmallinnus järjestelmä (kKuva 1), johon kuuluvat asiakassovellus prosessin määritykseen ja tiedostojen käsittelyyn, sovelluspalvelin, tietokanta, web-palvelin ja web-portaali prosessin katseluun.



Kuva 1: QPR Process Guide- järjestelmäkuvaus

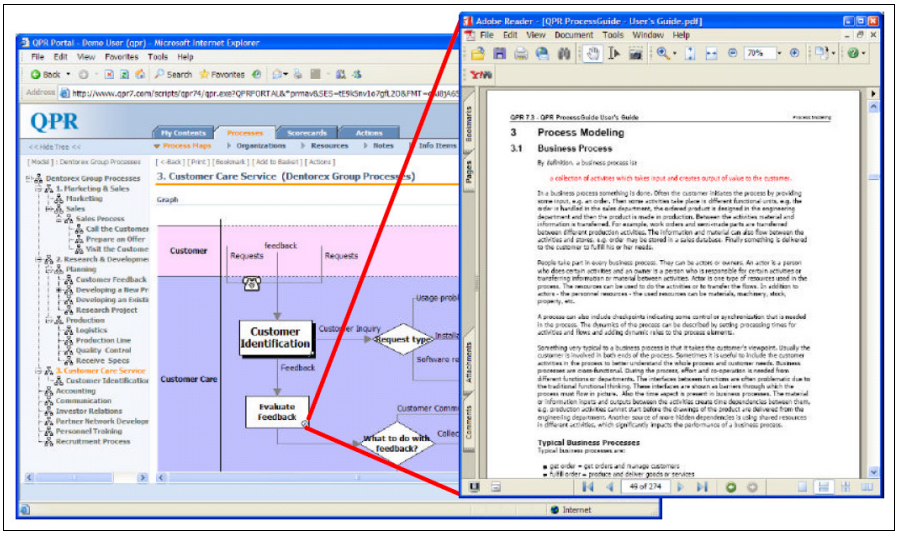
Prosessin aktiviteetit kuvataan graafisen käyttöliittymän (kKuva 2) avulla objekteiksi, joiden määrittelyt ja sisältö tallennetaan tietokantaan. [12.]



Kuva 2: Näkymä QPR Process Guiden prosessinmallinnuksen käyttöliittymästä

2.5.3 Prosessiportaali ja tuoteprojektin näkymä

Process Guide -ohjelmiston avulla tuoteprosessin kehitysryhmä tuottaa myös koosteen johon pääprosessikohtaiset etappivaatimuslistat liitetään tiedosto-objekteiksi. Tämän jälkeen tuoteprosessista tehdään julkaisu ja se asetetaan tuoteprojektien nähtäväksi QPR Process Portal -sovellukseen (kKuva 3), johon asiakkaat pääsevät suoraan web-selaimen avulla. [12].



Kuva 3: QPR Process Portal - prosessinäkymä ja linkitetty tiedosto

Process Portalista on tuotettu Nokialle räätälöity versio. Siinä on käytössä Nokian tyylimäärittelyt ja toimintoja on pelkistetty, jotta käyttöliittymä olisi selkeämpi. Tuoteprojektin henkilöt voivat selata portaalista koko tuoteprosessia, sen aliprosesseja, räätälöityjä prosesseja sekä kuhunkin prosessiin kuuluvien aktiviteettien toimintakuvauksia.

Prosessien etappiobjekteihin linkitetään julkaisun yhteydessä tuotevalmistusprosessin etappivaatimuslista joka sisältää kaikki prosessin vaatimukset kyseisestä valmistusvaiheesta. Objektin linkistä käyttäjä voi ladata vaatimuslistan MS Excel-tiedostona omalle koneelleen. [11.]

2.6 Prosessinhallintajärjestelmän tarve ja vaatimukset

Tuoteohjelmille tarkoitetun portaalin tarjoamien monipuolisten esitystoimintojen puitteissa ei ole mahdollista tehdä varsinaista prosessin käyttöönottoa, seurata prosessiaktiviteettien vaiheita ja raportoida etappivaatimusten toteutumista. Tästä syystä oli tuotettu aiemmin mainitut prosessin käyttöoppaat kullekin tuoteprojektityypille.

2.6.1 Toimitusvalmiusprosessin materiaalin muoto ja käsittely

Käyttöopas on MS PowerPoint -dokumentti, joka sisältää

- toimitusvalmiusprosessin yleisen tason kuvaukset kaavioina
- projektityypille olennaiset toimitusvalmiusprosessin aktiviteettien toimintakuvaukset MS Word-dokumentteina
- etappikohtaisesti jaettu lista prosessin aktiviteetteihin liittyvistä etappivaatimuksista.

Käyttöoppaan rinnalla ylläpidettiin toimitusvalmiusprosessin etappivaatimukset sisältävää MS Excel -raporttitiedostoa, jolla tuoteprojekteissa hoidettiin etappivaatimusten seuranta. Tuoteprojekteissa oli myös sallittua tehdä projektikohtaisia räätälöityjä prosesseja, jolloin oli tarve muokata projektille soveltuva manuaali ja etappivaatimusraportti. [11.]

Käyttöoppaiden sisältämät tiedot pääosin löytyvät jo aktiviteettien kuvauksista ja etappivaatimusten listauksista. Informaation monistumisen lisäksi käyttöoppaan päivitys vaatii kohtuuttoman paljon manuaalisia syötteitä sisällön päivitysten yhteydessä.

Toimitusvalmiusprosessi sisältää noin 120 aktiviteettikuvausta ja vastaavan määrän etappivaatimuksia määriteltynä kullekin tuoteprojektityypille. Tämän takia sähköisen materiaalin hallinnoiminen ja ylläpito on aikaa vievää ja raskasta. Siksi toimitusvalmiusprosessin kehitysryhmän tavoitteena oli lakkauttaa käyttöoppaan käyttäminen.

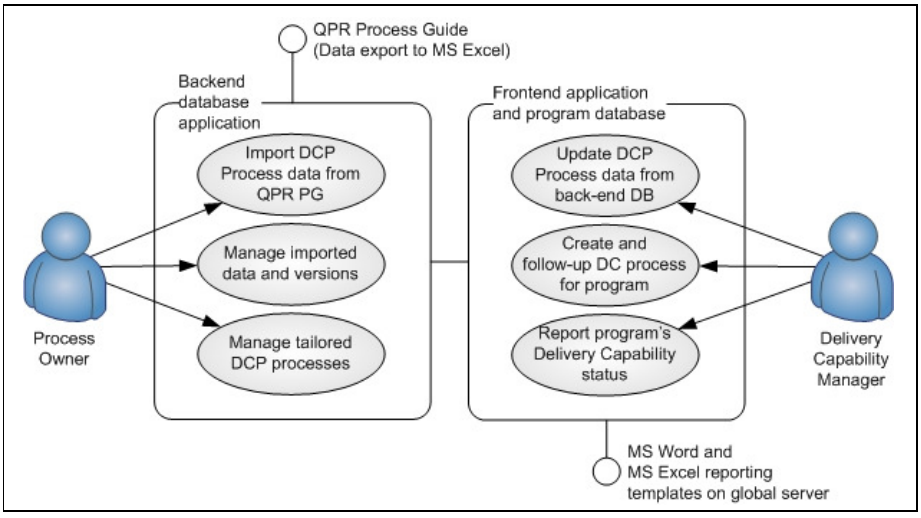
2.6.2 Kehitysvaatimukset

Tämän insinöörityön tavoitteena oli toteuttaa sovellus, joka kykenisi käyttämään hyväksi saatavilla olevaa informaatiota ja esittämään sitä tuoteprojekteille soveltuvassa muodossa ilman, että toimitusvalmiusprosessin kehitysryhmän tarvitsee käyttää vastaavaa resurssien määrää esitysten toteuttamiseen.

Insinöörityössä prosessinhallintaan toteutettavan ohjelmistoratkaisun olennaisena vaatimuksena oli, että se kykenisi hyödyntämään QPR-tietokannasta valmiiksi saatavaa tietoa, jotta manuaaliselta raporttien ja käyttöoppaiden päivitykseltä vältyttäisiin.

2.6.3 Toiminnalliset vaatimukset

Prosessinhallintaan toteutettavan sovelluksen toiminnallisten vaatimusten määrittämiseksi tutkittiin yritystoimintaa, nykyisen prosessin kehitysympäristön vaatimuksia ja haastateltiin loppukäyttäjiä yksityiskohtaisten toimintojen määrittämiseksi (kKuva 4).



Kuva 4: Prosessinhallintajärjestelmän vaaditut toiminnot ja liittymät

Tutkimusten perusteella toteutettavan sovelluksen tulee toteuttaa seuraavia toimintovaatimuksia:

- Prosessin kuvaus ja prosessin aktiviteettien kuvaus tulee tapahtua ainoastaan Process Guidessa.
- Prosessin etappivaatimusten kuvaus tulee tapahtua ainoastaan vaaditussa erillisessä MS Excel -tiedostossa.
- Sovelluksen tietojärjestelmän tulee pystyä hyödyntämään QPR Process Guide-järjestelmän tietokannan sisältöä.
- Prosessin hallinta täytyy tapahtua keskitetysti. Toteutettavan sovelluksen tuki sovelluksen hajautukselle ja tietokantayhteyksille täytyy varmistaa.
- Sovelluksen tulee sisältää tuoteprojektihenkilöstölle tarkoitetut käyttöliittymät, jotka mahdollistavat prosessin käyttöönoton ja seurannan.
- Työkalussa on otettava huomioon loppukäyttäjien tarpeet rakenteessa ja käyttöliittymän käytettävyydessä.
- Prosessia tulee olla mahdollista muokata pienillä muutoksilla usealle eri prosessille tai projektille sopivaksi työkalun pääkäyttäjän käyttöliittymästä käsin.
- Työkalusta tulee pystyä siirtämään tietoa toisiin sähköisiin järjestelmiin, kuten raportteja MS Exceliin.

2.6.4 Tekniset vaatimukset ja rajoitteet

Teknisten vaatimusten määrittämiseksi tutkittiin olemassa olevien toteutusten kustannuksia ja malleja sekä selvitettiin millaisia IT palveluita oli saatavilla. Sovellusratkaisun tutkimuksessa ja arkkitehtuurissa tuli huomioida seuraavia järjestelmälle asetettuja rajoituksia ja vaatimuksia:

- Sovelluksen tulee tukea joko tietokanta- tai tiedostopohjaista tietojen käsittelyä ohjelmallisesti.
- Sovelluksen tulee toimia tiedostopalvelinympäristössä. Sovellukselle ei ole tarjolla omia palvelinresursseja.
- Nokian työasemien standardiasennukseen kuuluu MS Office 2003 paketti ja sen sovelluksia on käytetty aiemmissa sovelluksissa.

QPR Process Guide järjestelmää koskien listattiin seuraavia vaatimuksia ja rajoitteita:

- QPR DB ja Process Guide tukevat prosessiobjektien vientiä ulkoiseen tiedostoon sekä ohjelmointia QPR API:n avulla, jolloin tiedostojen vientiä voidaan kontrolloida.
- QPR-makrojen käyttö vaatii ohjelmistolisenssin. Lisenssi on myönnetty prosessikehittäjälle.

Sovelluksen toimintaan liittyen suunnittelulle oli asetettu seuraavat tekniset vaatimukset:

- Valitun tietojärjestelmän täytyy kyetä toimimaan itsenäisesti (stand-alone), jotta se voisi toimia tiedostopalvelinympäristössä.
- Sovelluksen tulee sisältää käyttöliittymän tietokannan ja tiedonlatauksen hallintaan.
- Työkalun tulee tukea MS Officen tiedostoformaatteja vähintään raportointitoimintojen osalta.
- Ohjelmien tulee tukea usean käyttäjän samanaikaista sovelluksen suoritusta. Aiempien ohjelmistototeutusten MS Excel -formaatin ja MS Access -formaatin tiedostojen ja sovellusten tuki moniajolle täytyy tutkia.

2.6.5 Kustannusvaatimukset

Projektin kannalta kriittisenä ohjelmistokehityksen rajoitteena oli kustannustehokkuuden edellytys sekä projektin resursoinnissa että sovelluksen toteutuksen aikana. Projektiin varattiin yrityksen henkilöstöstä resursseiksi konsultit prosessinkehityksen ja siihen liittyvien työkalujen tutkimusta varten.

Ohjelmistoratkaisulle ja insinöörityöprojektille oli asetettu seuraavat kustannusvaatimukset:

- Tuotettavan ohjelmistoratkaisun tulee olla kustannustehokas.
- Projektiresurssit ovat rajalliset. Projektin henkilöstö osallistuu määrittelyyn ja testaukseen, mutta ohjelmointi tapahtuu täysin insinöörityön tekijän toimesta.
- Käyttökustannukset tulee minimoida.
- Ylimääräisten ohjelmistolisenssien mahdollinen tarve tulee kartoittaa ja perustella.

- Tuotettavalle ratkaisulle ei ole mahdollista käyttää erillisiä laitteistoja tai tehdä niitä koskevia hankintoja.

Projektissa tapahtuva ohjelmiston tuotanto ja siihen liittyvä tutkimus suunniteltiin toteutettavasi pelkästään projektihenkilöstön, pääasiassa insinöörityön tekijän toimesta.

3 MS OFFICE JA VBA-OHJELMOINTI

Sovellusarkkitehtuurin tutkimukset päätettiin perustaa MS Office -ohjelmien ominaisuuksiin ja Visual Basic for Applications -ohjelmoinnin tekniikoihin. Näiden rinnalla tehtiin vertailua vaihtoehtoisista järjestelmistä ja niiden ominaisuuksien eroista MS Office ja VBA-sovelluksiin nähden. Tähän rajaukseen päädyttiin, koska insinöörityön tekijällä ja projektiryhmällä oli eniten kokemusta VBA-sovellusten toteutuksista ja ohjelmointimenetelmistä.

MS Office on Microsoftin tuottama toimistosovellusten paketti, joka sisältää ohjelmia tekstin- ja tiedonkäsittelyyn (MS Word, MS Excel, MS Access), sähköpostin ja kalenterin hallintaan (MS Outlook) sekä sähköisten esitysten kuten diasarjojen (MS PowerPoint) ja internetsivujen tuottamiseen (MS FrontPage). MS Office on edelleen suosituin kaupallinen toimistosovellusten paketti sekä suurissa että pk-yrityksissä Windows- ja Mac OS X -käyttöjärjestelmille. [15, 16.]

Yrityksillä on monen tasoisia tiedonkäsittelytarpeita ja niitä vastaavat toteutukset vaihtelevat monimutkaisuudessaan ja määrässään. Tutkimuksessa perehdytään tiedonkäsittelyyn MS Excelissä ja MS Accessissa.

Vaikka MS Excel ei ole tietokantajärjestelmä, silti useiden yritysten organisaatioissa henkilöt tallentavat enemmän tietoja Exceliin kuin millekään muulle alustalle. Tähän on syynä se, että Excel tarjoaa päätöksentekijöille helppokäyttöiset työkalut tiedon analysointiin eikä sen käyttö vaadi ulkopuolisia resursseja. [7.]

MS Access taas tarjoaa alustan laajentaa taulukkopohjaiset tietovarastot luotettavampaan relaatiotietokantaratkaisuun. Sellaisia tiedonkäsittelytarpei-

ta, joihin MS Access soveltuu parhaiten, on yleensä huomattavasti enemmän kuin sellaisia, joihin soveltuu vain koko yrityksen laajuinen monimutkainen palvelinratkaisu. Tämä näkyy vastaavasti myös toteutusten kustannuksissa. [15, 16, 1, 7.]

3.1 MS Office -sovellukset ja niiden ohjelmointi

Microsoft listaa MS Office-paketin myös sovelluskehitysalustaksi jolla voidaan tuottaa kriittisiä yrityshallinnon sovelluksia (LOB, line-of-business software). Pääasialliset ohjelmointialustat MS Office paketissa ovat MS Excel, jossa VBA-ohjelmointia käytetään taulukoiden hallintaan ja laskutoimitusten suorittamiseen sekä MS Office Pro -versiosta löytyvä MS Access joka sisältää sovelluskehittimen tietokantasovellusten luomiseen. [7, 16.]

MS Officen ohjelmia on yhtenäistetty tukemaan Microsoft Visual Basic for Applications ohjelmointikieltä (MS Office 97 / VBA5.0 alkaen) sekä makroautomaatiota. Ohjelmointi antaa myös tuen OLE DB -dataintegraatiolle ja ActiveX-komponenttien käytölle. Lisäksi useimpiin Office-ohjelmiin on mahdollista asentaa käyttäjien tai kolmannen osapuolen tuottamia COM-lisäosia, joilla voidaan laajentaa ja räätälöidä ohjelman toimintoja. MS Office 2000 -versio toi mukanaan myös tuen VBA:n itse toteutettaville luokkaobjekteille. [7, 16.]

3.2 VBA-ohjelmointi

Seuraavaksi käsitellään tarkemmin VBA-sovellusten ominaisuuksia ja rakennetta sekä MS Officen integroitua VBA-kehitysympäristöä.

3.2.1 Visual Basic

Visual Basic (VB) on Microsoftin kehittämä tapahtumakeskeinen, helppokäyttöisyyteen ja nopeaan sovelluskehitykseen (RAD) tähtäävä Basic-sukuinen ohjelmointikieli. Visual Basic on suosittu Windows-sovellusten ohjelmointikieli, ja sillä on suurin osuus yritysohjelmoinnissa käytettävien kielten joukossa. [7, 1, 16.]

VB6.0 on tällä hetkellä laajin käytetty Visual Basicin versio, koska se on versioista parhaiten alaspäin yhteensopiva ja toimii vanhempien käyttöjärjestelmien kanssa. VB.NET on Visual Basicin päivitetty versio, joka tukee kaikkia .NET Framework -alustan luokkia.

Visual Basic ei ole aito oliokieli, mutta siinä on pitkälle vietyä olioiden simulointia, jolla saavutetaan useita olio-ohjelmoinnin hyötyjä (kuten koodin uudelleenkäytettävyys, luokkien käyttö, kapselointi, rajapinnat ja monistaminen) ja mahdollisuus käyttää olio-pohjaisia sovellusarkkitehtuureja. Visual Basic on kuitenkin korkean tason ohjelmointikieli ja Visual Basicilla useiden abstraktien asioiden käsittely on selkeätä ja yksinkertaista. [7, 1, 16.]

3.2.2 Visual Basic for Applications

Visual Basic for Applications on VB:n implementaatio, jota käytetään pääasiassa MS Office -ohjelmistoihin ja AutoCAD:iin integroidun kehitysympäristön (IDE) kautta OLE Automation -rajapinnan ohjelmointiin. Rajapinta mahdollistaa isäntäsovelluksen toimintojen sekä käyttöliittymän hallinnan makrosovelluksista ja sisältää liittymän suurimpaan osaan Windowsin järjestelmäkutsuista. Isäntäsovellus tarjoaa VBA-ohjelmalle myös laajan valikoiman erilaisia tapahtumapalveluita ja -käsittelijöitä joiden avulla VBA-ohjelman avulla voidaan ohjata käyttäjän toimintaa sovelluksessa. [7, 1, 16.]

Sovellusten kehittäminen VBA-kielellä on järkevä vaihtoehto laajempien VB6.0- tai VB.NET-kielten käytön sijaan, vaikka molemmat ovatkin kattavampia, kyvykkäämpiä ja suosittuja ohjelmointikieliä. Tästä huolimatta VBA:lla on kaksi tärkeää etua näihin nähden:

- VBA kuuluu MS Office -pakettiin
- VBA-ympäristö hyödyntää täysin MS Office -objektien ominaisuuksia.

VBA-ohjelmointikieli, VBA-objektikirjastot ja integroitu VBA-sovelluskehitin Visual Basic Editor on sisällytetty MS Office -pakettiin joka löytyy jo suurelta osalta yrityksistä, jolloin tuotettaessa VBA-sovelluksia säästytään ylimääräisiltä ohjelmoinnin lisäkustannuksilta. VB6.0:an tai VB.NET:in objektikirjastot kehittäjä joutuu hankkimaan joko erillisellä lisenssillä tai Visual Studio tai Visual Studio.NET sovelluskehittimen osana. Tämä tuottaa suuren kustannuseron, jos useampien kehittäjien täytyy päästä käsiksi VBA-kehitystyökaluihin.

Toiseksi, koska kaikki Office sovellukset tukevat VBA:ta, ohjelmoija voi hyödyntää niiden sisäänrakennettuja objektikirjastoja suoraan ohjelmakehityksessä ja varmistua siitä, että myös loppukäyttäjältä ne löytyvät. Lisäksi käytettäessä MS Officen sovelluskohtaisia objektikirjastoja VBA-ympäristössä ohjelmoija saa käyttöönsä objektien täydet ominaisuudet, jotka kattavat laa-

jan valikoiman sovelluksen toimintoja, sovelluskohtaisia tietorakenneobjekteja sekä useita käyttöliittymävalikoista löytyviä toimintoja. [1.]

3.2.3 Makroautomaatio ja toimenpiteiden nauhoittaminen

Makro on sarja sovelluksessa suoritettavia toimenpiteitä, joka voidaan tallentaa ja toistaa uudelleen. Kaikki Office-tuotteet tukevat makroautomaatiota, mikä mahdollistaa useiden rutiinitoimenpiteiden tallentamisen makroohjelmaksi. Rutiinitoimenpiteiden automatisointi parantaa resurssien käyttöä sekä vähentää ratkaisevasti virheiden määrää.

Yksinkertaisimmillaan käyttäjä tuottaa makroja MS Office -sovelluksissa nauhoittamalla Macro Recorder -työkalulla suorittamiaan käyttöliittymätoimenpiteitä, näppäinkomentoja ja syötteitä. Työkalu tulkkaa toimenpiteet automaattisesti VBA-kielisiksi aliohjelmiksi. Valmiin makron käyttäjä voi liittää näppäinyhdistelmään tai dokumenttiin sulautettuun nappiin. Nauhoitettujen toimintojen avulla käyttäjä voi luoda yksinkertaisia sovelluksia käytännössä ilman ohjelmointia. Usein kuitenkin on syytä räätälöidä nauhoitettuja rutiinioperaatioita uudelleenkäytettävyyden saavuttamiseksi tai käyttäjää ohjaavien lomakkeiden ja tapahtumakäsittelijöiden luomiseksi. Nauhoitettujen makrojen VBA-ohjelmakoodi tarjoaa tässäkin hyvän pohjan VBA-ohjelmien luomiseksi. [7.]

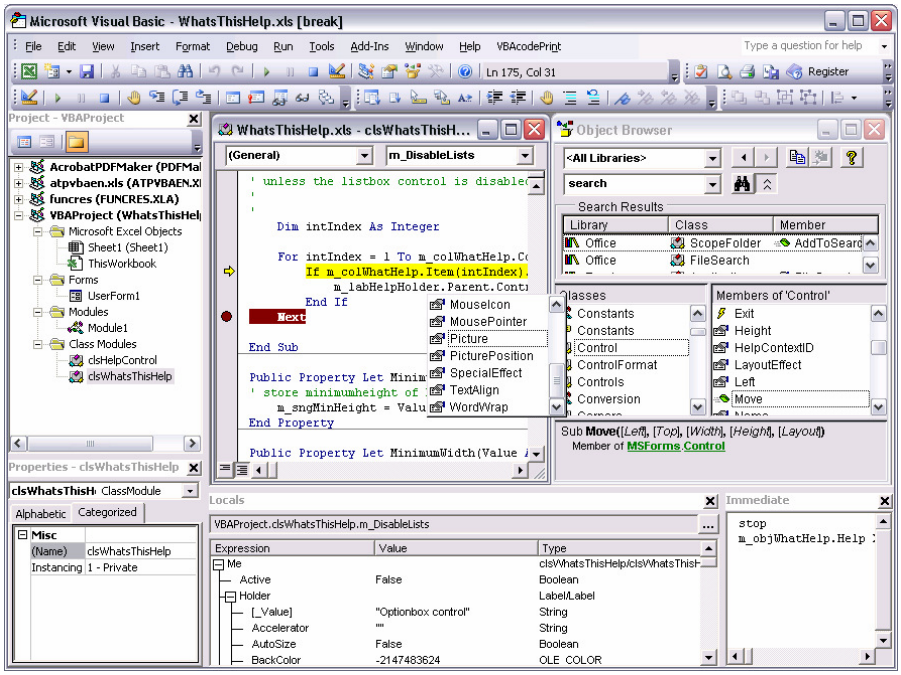
MS Access sisältää poikkeuksen makrojen käsittelyssä, koska siinä makroja hyödynnetään kehitettävän tietokantasovelluksen ohjelmointiin eikä kehitysvaiheessa yleensä ole tarvittavia toimintoja, joita voitaisiin nauhoittaa. MS Accessissa yksinkertaisia makroja tuotetaan erillisessä rakenneikkunassa tekstipohjaisesti. Toiminto tukee 40 erilaista valmista proseduuria, joille määritellään tietokantatietueisiin liittyviä attribuutteja ja käynnistykseen liittyviä tapahtumia.

Varsinaiset MS Access sovellukset, niiden tapahtumien käsittely ja automaatio tuotetaan kuitenkin käyttämällä Visual Basic for Applications -ohjelmointia. Silloin ohjelmoijan käytössä ovat kaikki VBA:n objektikirjaston objektit ja niiden ominaisuudet. [16, 1.]

3.2.4 Visual Basic Editor (VBE)

VBA-sovellusten kehitystä varten MS Office sisältää integroidun kehitysympäristön (IDE) Visual Basic Editor-ohjelman (kKuva 5), joka otettiin mukaan

MS Office 97 -versiossa. Ohjelma sisältää näkymät tiedostojen ja ohjelmaprojektin hallintaan, ohjelmakoodieditorin, valitun objektin ominaisuuksien esityksen, graafisen lomakekehittimen tapahtumaohjattujen lomakkeiden ja sovellusten luontiin, COM-objektikirjastoselaimen (Pelkästään julkisia objekteja MS Officessa on noin 500, ja niillä kaikilla ominaisuuksia ja metodeja) sekä toiminnot ohjelmansuorituksen virheenkäsittelyn seurantaan. MS Office sovelluksista Visual Basic Editor löytyy Tools-työkaluvalikon Macros-osasta (suomenkielisessä versiossa Työkalut / Makro). [7, 8.]



Kuva 5: Visual Basic Editor, MS Officeen integroitu VBA-sovelluskehitin

Visual Basic Editor sisältää kehittyneet työkalut ohjelmakoodin editoimiseen. Kehitin tarkistaa lennossa koodin syntaksin, värjää avainsanat, tarjoaa käskystä objektikirjaston perusteella syntaksivaihtoehtoja (esimerkiksi käsiteltävän objektin ominaisuudet ja metodit. Kts. kKuva 5, keskellä) sekä avustaa objektin ominaisuuksien määrittelyssä esimerkiksi listaamalla metodin vaatimat parametrit vihjetekstinä. [7, 16.]

Visual Basic Editorin heikkoutena mainittakoon automaattisen ohjelmakoodin jäsennyksen työkalujen puuttuminen, siitä johtuu, että VBA:n ohjelmakoodi on usein niukasti tai epästandardisti jäsennettyä ja koodin luettavuus kärsii. Asian voi kuitenkin korjata sovellukseen saatavilla kolmannen osapuolen COM-lisäosilla (kuten SmartVBA VBA CodePrint), jotka kykenevät

jäsennyksen lisäksi tuottamaan automaattisesti ohjelmapätkiä ja sovellusrakenteita sekä tuottamaan dokumentaatiota ja kaavioita sovelluksen ohjelmakoodista.

3.2.5 Ajonaikainen ohjelmakoodin tulkkaus ja virheiden käsittely

Visual Basic kuten myös VBA tukevat ohjelmakoodin ohjelman ajonaikaista tulkkausta. Visual Basic Editor tarjoaa laajan työkaluvalikoiman ajonaikaiseen virheiden paikannukseen. Tulkkauksen aikana ohjelmoijalla on mahdollisuus keskeyttää ohjelman ajo joko käskystä tai valmiiksi määriteltyyn keskeytyspisteeseen, tarkastella muuttujien arvoja ja jopa muuttaa niitä lennossa. [1.]

Sen lisäksi, että ominaisuus säästää runsaasti aikaa sovelluksen kääntämisessä ja virheiden tutkimisessa, tämä ominaisuus mahdollistaa VBA-ohjelmointikielen helpon omaksumisen ja opettelun lähes ilman ohjelmointitaitoja. Käyttäjä voi rivi kerralla koodia suorittaessaan välittömästi tutkia taustalla olevasta isäntäsovelluksesta suoritetun käskyn vaikutuksia ja tehdä korjauksia lennosta.

3.2.6 Visual Basic for Applications -sovellusten rakenne ja ominaisuudet

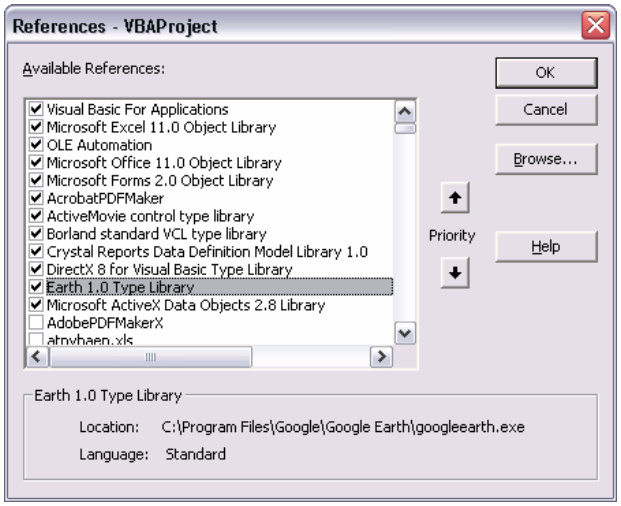
VBA-ohjelmat koostuvat makromoduuleista (Module), luokkamoduuleista (Class module), lomakkeista (UserForm) sekä niiden sisältämistä ohjelmakooditietueista eli proseduureista. [7.]

VBA:n proseduureja on kolmea tyyppiä:

- Komentomakrot eli sub-proseduurit (*Sub*) ovat yleisin tyyppi. Näitä käytetään myös nauhoitettujen makrojen tallentamiseen. Ne koostuvat yleisimmin lauseista, jotka käsittelevät ohjelman työkaluvalikoiden valintoja tai muita ohjelmakäskyjä. Sub-proseduurit kykenevät ottamaan vastaan objekteja proseduurikutsun argumenttien välityksellä.
- Käyttäjän määrittelemät funktiot eli funktio-proseduurit (*Function*), jotka ovat kykenevät ottamaan vastaan sekä palauttamaan objekteja funktiokutsun argumenttien välityksellä.
- Objektin arvojen asettamiseen ja palauttamiseen käytettävät ominaisuusproseduurit (*Property Get, Property Let*), jotka kykenevät muuttamaan luokkamoduulissa (*Class Module*) olevia suojattuja arvoja.

Objekteilla VBA:ssa voi lisäksi olla erilaisella suojaustasolla (*Public, Private, Const, Static*) määriteltyjä luokkaobjektin ominaisuuksia (*property*) tai proseduurin muuttujia (*variable*) sekä käyttäjän toiminnoista tai sovelluksen tiloista aktivoituvia tapahtumia (*Event*).

VBA-ympäristössä kaikki Office -ohjelmat julkistavat OLE Automaation avulla omat objektikirjastonsa VBA-sovelluksille. Tämä mahdollistaa esimerkiksi oliopohjaisesti toteutettavan MS Outlook-sähköpostin kirjoittamisen ja muotoilun MS Excelistä suoritettavasta VBA-ohjelmasta käsin.



Kuva 6: COM-objektikirjastojen tuonti VBA:han referenssinä

Visual Basic Editorin References työkalun avulla (kKuva 6) voidaan myös referoida toisen projektin sisällä olevia julkisia proseduureja tai luokkaobjekteja määrittämällä projektiin referenssejä. Referenssit voivat viitata toiseen tiedostotoon, joista löytyvää VBA-projektia voidaan referoida. Vaihtoehtoisesti voidaan referoida kokonaisia objektikirjastoja, kuten MS Forms 2.0, Microsoft ActiveX Data Objects 2.8 Library tai Google Earth 1.0 Type Library -kirjastot.Toista projektia referoimalla voidaan saavuttaa parempaa uudelleenkäytettävyyttä VBA-koodille, kun yleisesti käytettävien proseduurien ja luokkien sijainti vakioidaan yhteen paikkaan.

VBA käyttää oletuksena muuttujien elinkaarena proseduuritason laajuutta, mikä tarkoittaa, että muuttujat alustetaan ja luodaan proseduurin käynnistyessä paikallisiksi ja tuhotaan automaattisesti proseduurin suorituksen lopussa. Tämä tarkoittaa, että muuttujan arvo ei säily kutsukertojen välillä. Proseduuritasolla on mahdollista kuitenkin esitellä muuttujat staattisiksi (*Static*),

jolloin niiden elinkaarta jatketaan ja ne tuhotaan vasta kun moduulin proseduurien suoritus on päättynyt. Tällä tavalla voidaan toteuttaa esimerkiksi iteraatiota käyttäviä proseduureja joiden täytyy säilyttää edellisen suorituskerran tulos seuraavaa suorituskertaa varten.

Muuttujat voidaan myös määritellä moduulitasolle siirtämällä kaikille moduulin proseduureille tarjottavien muuttujien esittelyt moduulin alkuun listaksi. Nämä muuttujat alustetaan moduulin tarkastuksen yhteydessä kun jonkin moduulissa olevan proseduurin suoritus alkaa, muuttujan arvoa voi muokata mikä tahansa suoritettava moduuli, ja muuttuja tuhotaan kun moduulin proseduurien suoritus on päättynyt. Muuttujien ja proseduurien elinkaaren laajuutta ja siihen liittyviä toimenpiteitä tutkittiin suunnittelumallien yhteydessä, joista kerrotaan tarkemmin kappaleessa 7.2 VBA ohjelman hallittavuus, uudelleenkäytettävyys ja suunnittelumallit.

Jotta moduulitason muuttujiin ja proseduureihin voidaan referoida muista projekteista, ne tulee esitellä *Public*-tyyppisiksi. Julkisille rajapinnoille tulee määrittää uniikki nimi, joka poikkeaa muissa projekteissa käytettävistä muuttuja- ja proseduurinimistä. Tämä ei ole kuitenkaan pakollista, sillä muuttujaan viitatessa voidaan rajata erikseen käytettävä kirjasto ja objekti kuten esimerkiksi *Applications.ActiveWindow.WindowState*.

Oletuksena VBA-projektin esittelemättömät muuttujat alustetaan suoritusta varten tyyppiin *Variant*, joka voi sisältää minkä tahansa tyyppistä dataa kuten kokonaisia VBA:n objekteja. Toiminto nopeuttaa sovelluskehitystä, mutta voi nopeasti kerätä valtavan määrän tarpeettomia resursseja sovellusta suoritettaessa. Välitettäessä muuttujia argumentteina sub-proseduureille ja funktioille, muuttujat voidaan välittää joko kokonaisina objekteina (*byRef*), jolloin argumentin saanut ohjelma voi muokata lähetetyn objektin arvoja tai yksittäisinä arvoina (*byVal*), jolloin proseduuri saa vain objektin sisältämän arvon eikä proseduuri pääse muokkaamaan kyseistä objektia.

Kaikki muuttujat VBA-ohjelmissa on syytä supistaa esittelyn avulla vain tarpeellisen kokoiseksi (esim. *Integer*-tyypille tarvitsee varata muistia vain 2 bittiä) sekä käyttää moduuleissa *Option Explicit* -määritettä, joka käskee Visual Basic Editorin kääntäjää tarkistamaan, että kaikille muuttujille löytyy esittely. Tehokkaaseen muuttujien esittelyyn ja muuttujien tyyppien selvittämisen helpottamiseksi on syytä käyttää sovittuja nimeämiskäytäntöjä. MS Acces-

sissa yleisin nimeämiskäytäntö on LNC (Leszynski naming convention), jossa käytetään lyhyitä etuliitteitä muuttujien määrittelyyn (esim. *str_*-etuliite String- ja *frm_*-etuliite UserForm-tyyppien määrittelyssä). Muuttujien laajuuden ja referoitujen kirjastojen minimoimisella saavutetaan pienempiä VBA projekteja ja suorituskykyisempiä VBA ohjelmia. [7.]

3.2.7 Soveltuvuus ja rajoitteet

Reunaehtona VBA-sovelluksen käytölle on aina isäntäohjelman käyttö ja olemassaolo. Se tarjoaa alustan, missä itse VBA-sovellus tulkitaan ja suoritetaan. Tässä projektissa tavoite on käyttää MS Office 2003 Pro ohjelmistopaketin tuotteita isäntäohjelmana. [7.]

Mikäli käyttäjällä ei ole lisenssiä MS Office ohjelmistoon tai vastaavaan VBA sovelluskehittimen sisältämään ohjelmistoon, on syytä tutkia avoimen lähdekoodin ohjelmointikieliä ja -tuotteita kuten Javaa ja MySQL-tietokantajärjestelmää. Avoimen lähdekoodin kehitysalustat ovat ilmaisia ja niille löytyy usein ilmaisia työkaluja myös ohjelmistokehitykseen. Jos lisäksi on tarkoitus investoida sovelluskehittimiin ja palvelimiin, tulee valikoimaan paljon muitakin vaihtoehtoja kuten C++-, VB.NET-sovellukset, MS SQL- ja Oracle-tietokantapalvelimet ja niin edelleen. [1, 5, 4.]

Kuten millä tahansa yleisellä ohjelmointikielellä, myös VBA:lla on mahdollisuus tuottaa makroja jotka aiheuttavat haittaa käyttäjälle. MS Office sisältää Internet Explorer -tyyliset suojaustason määrittelyt ja oletuksena käyttäjältä vaaditaan hyväksyntä VBA-sovellusten suorittamiseen isäntäohjelmassa. Tiukin turva-asetus kuitenkin estää makrojen suorituksen kokonaan, jolloin hyväksyntää ei kysytä eikä VBA-sovellusta voida ajaa käyttäjän koneella ilman turva-asetusten uudelleenmäärittelyä. [15.]

Korkean tason kielenä Visual Basic on ilmaisuvoimaltaan melko rajoittunut - laitteistotasoon on vaikea päästä suoraan käsiksi. Tässä projektissa laitteistotason toiminnot eivät ole oleellisessa asemassa. Vastaavasti korkean tason arkkitehtuurista on tälle projektille hyötyä, kun useiden abstraktien asioiden käsittely on VBA-kielessä selkeätä ja yksinkertaista. [7.]

Visual Basic ja siten VBA ei ole aito oliokieli, mutta siinä on pitkälle vietyä olioiden simulointia, jolla saavutetaan useita olio-ohjelmoinnin hyötyjä ja mahdollisuus käyttää olio-pohjaisia sovellusarkkitehtuureja. Visual Basicin

päivitys Visual Basic .NET sisältää useita arkkitehtuurimuutoksia kielen rakenteessa, ja VB.NET toteuttaa täysin oliokielen määritelmän. Tämä kuitenkin aiheuttaa sen, että VB6.0- tai VBA-sovellukset eivät ole yhteensopivia .NET-sovellusten kanssa. VBA:sta on myös toteutettu päivitys Visual Studio Tools for Applications (VSTA) joka perustuu .NET-arkkitehtuuriin ja on julkaistu MS Office 2007 -version kanssa. Tämä työ ei kuitenkaan perehdy .NET-arkkitehtuurin ominaisuuksiin, koska käytettävä MS Office 2003 -versio perustuu edelleen COM-arkkitehtuuriin perustuvaan VBA6.0:aan.

Visual Basicia on myös moitittu heikosta muistinhallinasta ja vähäisestä ohjelmoinnin standardoinnista. Usein on vaikeaa tunnistaa, onko muuttuja referenssi objektiin vai itse eksplisiittinen objekti. Samalla on vaikea määrittää palauttaako operaatio kopion objektireferenssistä vai kokonaisen kopion itse objektista. Tämä epäselvyys voi johtaa odottamattomiin sovelluksen toimintoihin tai johtaa sovelluksen heikkoon suorituskykyyn. Tässä projektissa ongelmaan on pyritty puuttumaan käyttämällä lähdemateriaalin tarjoamaa tyylimäärittelyä ohjelmoinnissa sekä tutkimalla yksittäisten funktioiden käsittelemien objektien kokoa sekä ajonaikaisen muistin varausta. [7, 16, 1.]

4 VAIHTOEHTOISTEN JÄRJESTELMIEN KÄYTTÖ

MS Office -tuotteiden käytön vaihtoehtoina tutkittiin laajempien ohjelmointikielten VB6.0:n, VB.NET:n, C++:n ja Javan käyttöä sovelluksen kehityksessä sekä avoimeen lähdekoodiin perustuvan MySQL-tietokantajärjestelmän ominaisuuksia.

4.1 Itsenäiset Visual Basic 6.0- ja Visual Basic.NET -sovellukset

VB6.0- ja VB.NET -sovellukset voidaan kääntää siten, että niitä voidaan käyttää erilaisilla tietokoneilla ja erilaisilla käyttöjärjestelmillä, kun sovellukseen sisällytetään kääntämisen yhteydessä sen tarvitsemat ajonaikaiset kirjastot eli Windows DLL-tiedostot. [5.]

Suunnitelmana oli käyttää MS Officeen kuuluvan MS Accessin tietokantatoimintoja Toimitusvalmiusprosessin datan käsittelyyn. Sovellus oli tarkoitus

tuottaa vain Windows-käyttöjärjestelmille ja Nokian työasemien standardiasennukseen kuului jo MS Office Pro. Näistä ja aiemmin mainituista kustannusvaikutuksista johtuen VB6.0 ja VB.NET koettiin tarpeettomiksi tämän projektin kannalta.

Lähdemateriaalista kävi kuitenkin ilmi, että sovelluksen VBA:lla tehtyjä moduuleja, luokkia ja komponentteja oli mahdollista tuoda suoraan Visual Studio-sovelluskehittimellä osaksi VB6.0-sovelluksia. VBA-moduuleja oli mahdollista tuoda myös VB.NET-sovelluksiin joillakin muutoksilla. Katsottiin, että laajentamismahdollisuudet VBA-sovelluksesta suurempaan järjestelmään olivat riittävät. [7.]

4.2 C++--sovellukset

C++--sovellukset tukevat täysin COM-arkkitehtuuria, mikä mahdollistaa suoraan MS Office -tiedostojen kuten MS Excel -taulukoiden, MS Word -dokumenttien ja MS Access -tietokantojen käytön C++-sovelluksissa esimerkiksi tietolähteinä ja raportointityökaluina. C++-sovellukset tukevat myös täysin ODBC-rajapintaa, jolloin voidaan käyttää mitä tahansa valinnaista tietokantajärjestelmää kuten MS Accessia, MS SQL Serveriä tai Oraclea. [13.]

Graafisten C++-sovellusten raskaan kehityksen ja sovellusten vaikeaselkoisuuden vuoksi, ja koska tutkimuksen hetkellä ei ollut tuntemusta ilmaisista C++-kehitysympäristöistä, päätettiin että C++ ei ole soveltuva tämän projektin käyttöön.

4.3 Java-sovellukset ja COM-objektirajapinnan käyttö

Avoimeen lähdekoodiin perustuva ja vapaasti jaettava Java-alusta on myös vaihtoehtoinen arkkitehtuuri VBA:n rinnalla. Java tukee tietokantayhteyksiä pääasiassa ODBC ja JDBC-rajapintojen kautta, jolloin voidaan käyttää vapaasti valittavaa tietokantajärjestelmää. Tavoitteena oli tuottaa asiakassovellus tietokannan rinnalle, jolloin tulisi käyttää Javan Swing-arkkitehtuuria, jolla voidaan tuottaa graafisia tapahtuma-ajoisia käyttöliittymiä. [13.]

Java-arkkitehtuuri ei tue COM-arkkitehtuuria ja MS Officen tiedostoformaateja suoraan, mutta sille on tuotettu erillisiä Microsoftin OLE 2 Compound Document formaattiin perustuvia rajapintoja, jotka mahdollistavat vastaavasti

MS Office -tiedostojen kuten MS Excel -taulukoiden, MS Word dokumenttien ja MS Access tietokantojen käytön Java sovelluksissa.

Tällaisia rajapintoja ovat

- *The Apache POI Project*, joka on API-kokoelma, joka mahdollistaa MS OLE 2 Compound Document formaatin tiedostojen manipuloinnin ja MFC kirjaston objektien sarjallistamisen puhtaasti Java-kieltä käyttäen. [14.]
- *J-Integra*, joka on välitason yhdysohjelma (Middleware), joka mahdollistaa Java ja COM-arkkitehtuurin yhteen toimivuuden DCOM protokollan Java-implementaation avulla. Sovelluksesta on tarjolla 30-päivän ilmainen kokeiluversio. [2.]
- *Jackcess Project*, joka on MS Access MDB-tiedostojen manipulointiin erikoistunut Java API kirjasto ja käyttää JDBC-rajapintaa. [3.]

Avoimeen lähdekoodiin perustuvien COM-rajapintojen implementaatioiden esittelyt ja koulutusmateriaalit olivat tutkimuksen hetkellä melko vähäisiä, joten API:n käytön katsottiin hidastavan merkittävästi projektin etenemistä. Lisäksi Java-sovelluksista päätettiin luopua tässä sovelluksessa, koska Javan Swing-pohjaisista tietokantaa käyttävistä asiakasohjelmista ei ollut kattavaa ohjelmointikokemusta.

4.4 MySQL-tietokantasovellukset

MySQL on suosittu ja tehokas avoimeen lähdekoodiin perustuva tietokantahallintajärjestelmä, jota kehittää ruotsalainen MySQL AB. Yritys tarjoaa tuotteesta sekä kaupallisia graafisilla työkaluilla varustettuja lisenssejä, että ilmaista GPL-lisenssiin perustuvaa *MySQL Community Edition*-jakelua.

MySQL kantaa käytetään pääasiassa kolmitasoarkkitehtuurissa palvelinsovelluksena, mutta se on myös mahdollista asentaa itsenäiseksi tietokannaksi, jolloin palveluita integroidaan osaksi tietokantasovellusta. Tämä vaatii VBA-sovellukselta Windowsin rekisterin muokkaustoimenpiteitä, jotta voidaan tallentaa tarvittavien MySQL-kirjastojen sijainnit. Tällä tavalla tulisi saavuttaa vaatimuksen mukainen tiedostopalvelinympäristössä toimiva tietokantajärjestelmän konfiguraatio. [10, 9, 7.]

Koska projektin jäsenillä ei ollut kattavaa tuntemusta MySQL tietokannan hallinnoinnista ja Windows rekisterin manipuloinnista VBA sovelluksilla, pää-

tettiin että MySQL:n sijaan käytetään MS Accessia, joka täyttää ympäristön asettamat vaatimukset itsenäisenä tietokannanhallintajärjestelmänä.

5 VBA-LOMAKEOHJELMOINTI

Yhtenä arkkitehtuurivaihtoehtona ohjelmistoratkaisulle tutkittiin tapahtumaajoisia lomakkeita hyödyntävää VBA-sovellusta, joka toimisi MS Excelissä. Kappaleessa käsitellään sovelluksen arkkitehtuurin ominaisuuksia ja sen soveltuvuutta projektin ohjelmistoratkaisuun.

5.1 MS Forms -komponenttikirjasto ja muut lomakekirjastot

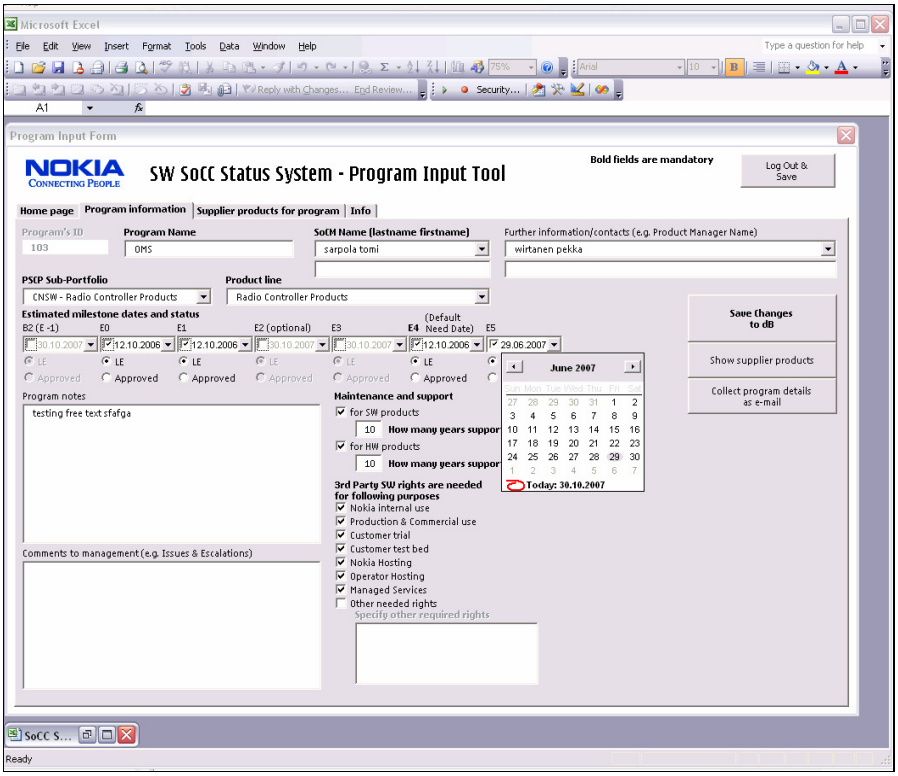
VBA sisältää laajan valikoiman erilaisia Windowsin graafisia komponentteja yksinkertaisista ponnahdusviesteistä aina mittaviin ikkunointi-, välilehti- ja toimintokomponentteihin sekä Officen käyttöliittymäkomponentteihin asti. Kaikki nämä ovat saatavilla Visual Basic Editorissa lomake-objektien kautta (kKuva 5). Kirjastot (kKuva 6) otetaan käyttöön referoimalla ne mukaan projektiin, lisäksi Visual Basic Editorin Toolbox-näkymään tulee lisätä tarvittavat komponentit Additional Controls -toiminnon kautta. [7.]

VBA tarjoaa sekä isäntäsovellukselle että kaikille graafisille komponenteille kattavan valikoiman tapahtumia. Tapahtuman kuuntelijat on sidottu aina yksittäiseen lomaketiedostoon, johon kuvataan sekä komponenttien sijoittelu että ohjelmakoodi. [7.]

5.2 Tiedonkäsittelysovellus MS Excelillä ja VBA-lomakeohjelmointi

Vuoden 2006 kesän aikana toimin Nokia Networksilla ohjelmoijana päivitysprojektissa, jossa MS Excel-pohjaiseen tietojärjestelmään toteutettiin VBA-sovellus tiedonkäsittelyä varten. Alkuperäinen tiedonkäsittelysovellus oli toteutettu Excelin tiedonkäsittelykaavojen ja valintalistatoimintojen avulla. Tiedonkäsittelysovellukseen tuotettiin VBA:lla graafinen käyttöliittymä (kKuva 7) sekä hajautettiin sovellus erillisiksi tietokanta- ja asiakassovellusmoduuleiksi tiedostopalvelinympäristöön. Tiedonkäsittelysovellus sisälsi toiminnot joilla Excel-taulukoista luettiin VBA:n avulla ohjelman lomakkeille rivikohtaista

muokattavaa tietoa sekä ladattiin vakiosyötteitä sisältävistä taulukoista listoja lomakkeiden komponentteihin kuten alasvetovalikoihin ja taulukkoesityksiin.



Kuva 7: Näkymä MS Excel VBA-sovelluksesta, joka hyödyntää lomakkeita.

Sovelluksen uudelleenkäytettäviä proseduureja oli esimerkiksi valintalistojen käsittelyä varten toteutetut proseduurit, jotka lukivat Excel-taulukkoa vastaavasta *Sheets.Range* -objektista listan sisällön ja täyttivät tiedot käsiteltävään valintalistaan riveiksi listakomponentin *AddItem*-metodilla. Proseduuri saatiin yleispäteväksi viemällä se erilliseen makromoduuliin ja asettamalla sille kutsurajapintaan attribuutit lomakemoduulin viittausta, lomakkeen komponentin viittausta ja lähteenä toimivan *Sheets.Range*-objektin viittausta varten. Lisäksi proseduuriin lisättiin lajittelumakro, joka käy Quick Sort -lajittelumenetelmällä listan alkiot läpi ja järjestää ne aakkosjärjestykseen.

Lomakkeen tallennusoperaatioihin toteutettiin vastaavasti uudelleenkäytettäviä proseduureja, jotka hakivat käsiteltävän rivin taulukoista lomakkeelle ja tallensi lomakkeella tehdyt rivikohtaiset muutokset takaisin Excel-taulukkoon. Tämä onnistui yksinkertaisesti käyttämällä VBA-sovelluksessa hyväksi Excelin sisäänrakennettuja *Find* ja *Replace*-metodeja, jotka olivat kutsuttavissa *Worksheets.Range*-luokasta.

5.3 Arkkitehtuurin soveltuvuus ja rajoitteet

Työn ohjelmistoratkaisun vaatimuksina oli Excel-datatulosteiden lukemiskyky, tietokantatoiminnot ja ylläpitoon tarvittavan käyttöliittymän toteutus. MS Excelissä tuotetut VBA-sovellukset kykenevät vastaamaan näihin vaatimuksiin. Lomakeohjelmointi ja käyttöliittymäautomaation ohjelmointi on varsin yksinkertaista ja nopeaa toteuttaa VBA:lla.

6 ULKOISET TIETOLÄHTEET JA VBA-SOVELLUKSEN HAJAUTTAMINEN

VBA-tarjoaa Office-tuotteiden avulla kattavat ja helppokäyttöiset toiminnot tiedonkäsittelyyn ja raportointityökalujen tuottamiseen. Tietolähteinä VBA-sovelluksissa voi käyttää tekstipohjaisia- tai Office-formaatin objektipohjaisiatiedostoja tai ulkoisia tietolähteitä jotka tukevat SQL-kyselykieltä.

6.1 Tiedosto-operaatiot VBA:lla

Mikäli VBA-sovelluksessa tarvitsee käsitellä tiedostoja, kuten edellisen kappaleen sovelluksessa, VBA tarjoaa laajan valikoiman valmiita yksinkertaisia proseduureja tiedostojen käsittelyyn. Toimintoihin kuuluvat tiedostojärjestelmän toimenpiteet kuten *CurDir, Dir, ChDir, MkDir, FileCopy, Name, Kill, RmDir, SetAttr,* graafiset dialogit *Applications.GetSaveAsFilename* ja *Workbooks.Open*. Lisäksi VBA sisältää matalan tason I/O-operaatioita mahdollistavia komentoja, jotka nostavat esiin joitain käyttöjärjestelmätason tiedosto-operaatioita, kuten tiedoston jaksotetun käsittelyn (*Sequential File Access*), hajasaantikäsittelyn (*Random File Access*) ja binääritason käsittelyn toiminnot. [7.]

Tiedosto-operaatioiden avulla aiemmin esitettyyn MS Excel -pohjaiseen tietojenkäsittelysovellukseen tuotettiin toiminnot, jotka päivittivät asiakkaan välimuistiin tallennetun muokatun rivin sisällön keskitettyyn tietokantatiedostoon. Automatisoidun tiedoston käsittelyn ansiosta tietokantatiedoston käsittelyaika saatiin minimoitua ja tiedosto oli lukittuna käyttäjälle vain hyvin lyhyen ajan. Lisäksi järjestelmään toteutettiin yksinkertainen rivikohtainen lukitusjärjestelmä joka esti päällekkäiset päivitykset. Näin toteutettu asiakas-

palvelin -sovellus oli kevyt toteuttaa, toimi ilman palvelimella suoritettavaa logiikkaa ja kykeni palvelemaan useaa käyttäjää samanaikaisesti.

Tiedostojen käsittelymenetelmät ja Excelin kyky toimia taulukoidun tiedon kanssa sopivat hyvin jäsennettyjen tietojen käsittelyyn, mutta suurten tietomäärien kanssa tulee hyödyllisemmäksi käyttää relaatiotietokantaa. [5.]

6.2 MS Queryn ja ulkoisten tietolähteiden hyödyntäminen

Edellisessä kappaleessa käsitellyn sovelluksen raportointi oli toteutettu MS Query:n avulla. MS Query on MS Exceliin integroitu graafisella käyttöliittymällä varustettu SQL-kyselykieltä tukeva sovellus, jolla kyselyn tulokset voidaan suoraan tuoda MS Exceliin taulukoiksi. MS Query-kysely tallentuu tiedon tuonnin yhteydessä Excel-välilehdelle, ja tämän jälkeen tiedon päivittäminen kaikille työkirjan välilehdille onnistuu VBA-ohjelmassa yhdellä komennolla: *ThisWorkbook.RefreshAll*. [7.]

Raporteille pystytään luomaan myös kyselyjä ohjelmallisesti. Tietojenkäsittelysovelluksessa käytettiin mallia, jossa valmiiksi määritetty MS Query -haku tuottaa tiedot pelkästään raportilla olevan valintalistan sisältöä varten ja valinnasta aktivoituva VBA-makro tuottaa valitun tietorivin perusteella rajatun haun. Haku luodaan *ActiveSheet.QueryTables* -objektin *Add*-metodilla, joka tarvitsee parametrina yhteystyypin ja tietolähteen polun. Luotuun hakuun liitetään haluttu SQL-kysely *String* -tyyppisenä parametrina *ActiveSheet.QueryTables* -objektin *CommandText*-ominaisuuden kautta. [7.]

Edellä käsitellylle tietojenkäsittelysovellukselle oli toteutettu arkkitehtuurin hajautus näitä menetelmiä käyttäen. Sovellukseen kyettiin siten kirjoittamaan syötettä useamman käyttäjän koneelta tiedostopalvelimella sijaitsevaan keskustiedostoon. Sovelluksen eri yritystoimintoja koskevat syötelomakkeet oli hajautettu omiin tiedostoihin ja niiden välissä sijaitsi keskustiedosto, jossa ylläpidettiin molempia syötteitä.

Tätä toimintaa varten keskustiedostoon toteutettu taulukkorakenne kopioitiin syötelomaketiedostojen välilehdille, jotka toimivat tiedon väliaikaistallentimina. Välilehtiä päivitettiin MS Queryn avulla staattisia hakuja käyttäen. Käyttäjät pystyivät tällä menettelyllä lukemaan ajantasaista tietoa ja tekemään sen perusteella muutoksia, jotka päivitettiin VBA-ohjelmalla keskustiedostoon. Sovellus avaa keskustiedoston VBA:n avulla tallennusoperaatioissa vain ly-

hyeksi hetkeksi, jolloin muutokset kirjoitetaan taulukoihin. Tämän jälkeen keskustietokannan tiedot luetaan uudestaan syötetiedoston taulukoihin päivittämällä tiedostoon tallennetut MS Query haut.

6.3 Laskentasovellus MS Excelissä ja toimintojen hajauttaminen

Osallistuin myös Nokia Networksilla vuoden 2006 aikana päivitysprojektiin, jossa MS Exceliin toteutettuun tuotteen toimituksen elinkaarikustannuksia mallintavaan laskentasovellukseen tuotettiin päivitystä.

Sovelluksen julkaistun version käyttäjiltä saadun vaatimuksen mukaisesti tutkittiin eri malleja, joilla koko laskentasovelluksen sisältävän yksittäisen Excel-tiedoston massiivista kokoa (5 MB) voitaisiin rajoittaa ja siten keventää loppukäyttäjälle kerääntyvää tiedostojen kuormaa. Se aiheutui, kun kutakin laskelmaa varten tarvittavat syötteet täytyi tallentaa laskentasovelluksen sisältämään tiedostoon, jolloin laskentatiedostoa monistettiin tarpeettomasti.

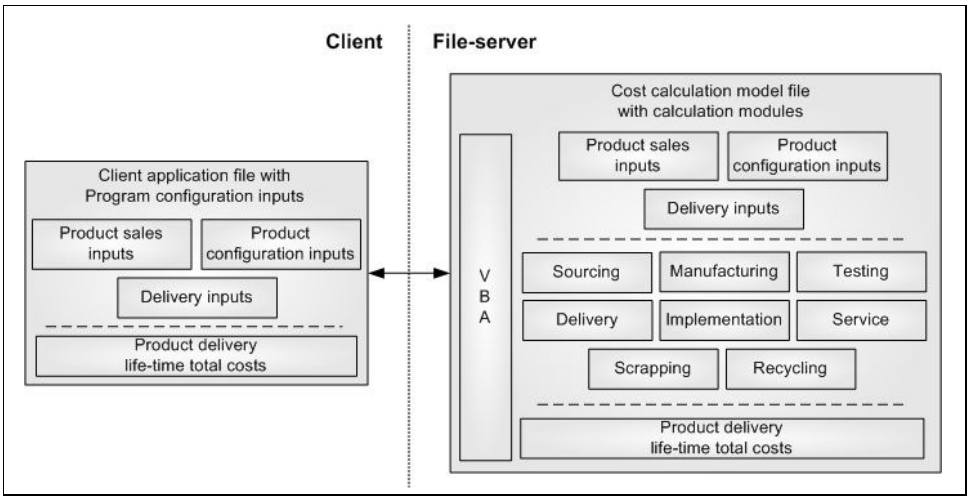
Sovelluksen päätoiminnot

Käyttäjä syöttää sovellukseen laskettavan tuotteen tiedot, kuten tuotantovolyymin, lopputuotteen painon, osien määrän, komponenttien määrän ja muita toimituksen kokonaiskustannukseen vaikuttavia tekijöitä. Sovellus laskee käyttäjän syötteestä jaetun kustannusmallinnuksen moduuleissa parametrien vaikutuksen toimituskustannuksiin, jotka koostuvat hankinta-, tuotanto-, testaus-, toimitus-, asennus- ja ylläpito- sekä purku- ja kierrätyskustannuksista. Tuotanto- ja ylläpitoajan perusteella kustannukset jaetaan aikajaksoihin ja lasketaan koko tuotteen elinkaaren ajalle.

Tuotteen toimitusparametreja ja komponenttikonfiguraatiota muokkaamalla käyttäjä tuottaa kokonaiskustannuslaskelmia, joita vertaamalla valitaan kannattavin tuote- ja toimitusmalli.

Hajautusmenetelmä laskentasovelluksessa

Laskentasovelluksesta toteutettiin hajautettu VBA-sovellus (kKuva 8), jonka avulla tuotekohtaiset syötetaulukoiden koko saatiin pienennettyä 90 % ja arkistointiin tarvittavat resurssit minimoitua.



Kuva 8: Laskentasovelluksen hajauttaminen tiedostopalvelinympäristössä

Asiakkaan tiedostosta käynnistettävä VBA-sovellus avaa laskentamallitiedoston tiedostopalvelimelta asiakkaan työasemalle ja määrittää avatun työkirjan muuttujaksi *Workbooks.Open*-metodilla:

```
Set wkbCalc = Workbooks.Open(str_FilePath + str_FileName, _
       UpdateLinks:=False, ReadOnly:=True)
```

Tämän jälkeen *wkbCalc*-objektin kautta voidaan kutsua tiedostossa olevia laskentamallia kontrolloivia VBA-ohjelmia *Call*-metodin avulla:

```
Call wkbCalc.AppModuleClass. _
       InputFile_CalcBtn_Clicked(str_CallerWbkName)
```

Kutsuttu *InputFile_CalcBtn_Clicked*-ohjelma siirtää tuotteen tiedot asiakastiedostosta laskentamallitiedoston syötekenttiin. VBA-ohjelma etsii ohjelman argumenttina saadun tiedostonimen (*str_CallerWbkName*) perusteella isäntäsovelluksesta ohjelmaa kutsuneen asiakastiedoston. Ohjelma asettaa viitteet sovittuihin solusijainteihin, ja lukemalla solusijaintien tietosisällöt laskentamallitiedoston vastaavasti sovittuihin solusijainteihin joko massa-ajona tai solu kerrallaan mikäli on tarvetta tutkia tietosisältöä:

```
Set r_main_inputs = Workbooks(str_inputWbkName).Sheets(2) _
  Range("B1:I250")
r_main_dest.Value = r_main_input.Value
```

Tämän jälkeen ohjelma suorittaa laskentamoduulien kaavojen päivityksen sekä laskentamakrot ja päivittää tulokset raporttisivun taulukkoon. Ohjelman alkaessa laskentasovellustiedostoon asetetaan laskentakaavojen päivityksen esto, jotta sovelluksen suoritus nopeutuisi ja syötteiden kopioinnin aikana ei turhaan laskettaisi kaavoja uudelleen. Tämä tapahtuu asettamalla *Ap-*

plications.Calculation-ominaisuus arvoon *xlCalculationManual*. Vastaavasti syötteiden kopioinnin jälkeen kaavat päivitetään asettamalla ominaisuus takaisin arvoon *xlCalculationAutomatic*.

Lopuksi ohjelma kopioi tulostaulukot asiakkaan tiedostoon luettavaksi ja tallentaa käyttäjän tiedot ja käytetyt parametrit suojattuun lokiin. Asiakas voi halutessaan jättää myös syötteet sisältävän laskentatiedoston avoimeksi näytölle, jolloin voidaan perehtyä tarkemmin laskentamallin toimintaan ja parametrien vaikutukseen.

6.4 Hajautetun moniajoympäristön tuki MS Excelissä

Työn ohjelmistoratkaisun vaatimuksina oli toteuttaa asiakassovellus tietokantatoimintoja tukevalle arkkitehtuurille. Kuten esimerkeistä havaitaan MS Excelissä tuotetut VBA-sovelluksen hajautus palvelin- ja asiakassovelluksiin on varsin yksinkertaista. VBA tarjoaa toiminnot tiedostojen käsittelyyn ja MS Queryllä voidaan nopeasti toteuttaa uudelleenkäytettäviä hakuja eri tietolähteistä. Ohjelmakoodia on mahdollista suorittaa myös palvelimella olevasta tiedostosta, jolloin hajautuksen ansiosta ohjelmakoodin hallinta ja päivitys asiakkaille yksinkertaistuu.

Insinöörityössä tuotettavan järjestelmän vaatimuksena oli myös tuki usean käyttäjän samanaikaiselle suoritukselle. Excel-pohjainen tiedostojen käsittely asettaa rajoitteet useiden käyttäjien tuelle, koska Excel-tiedosto ei tarjoa kuin yhden yhteyden tiedostoon kerrallaan. Tämä riittää vielä sovelluksessa, jossa transaktioiden tiheys ei ole suuri ja tallennusproseduurit ovat lyhytkestoisia. Lisäksi toimenpiteitä voidaan nopeuttaa alhaisentason I/O operaatioita käyttämällä. Tämä taas monimutkaistaa ja jäykistää sovelluksen rakennetta. Ensimmäisen esimerkin tietojenkäsittelysovelluksessa tiedostoyhteys otetaan käyttöön joka kerta, kun tiedostoon kirjataan muutoksia. Lukeminen tulisi tapahtua MS Queryn avulla *read-only* -tilassa, mutta toimintoa ei saatu toimimaan oletettavasti käytössä olleen ODBC-yhteysmäärittelyn takia.

7 MS ACCESS -TIETOKANTASOVELLUKSET

Insinöörityön tavoitteena oli prosessinhallintatyökalun suunnittelu ja toteutus toimitusvalmiusprosessin kehitysryhmälle. Sovellus päätettiin toteuttaa MS Accessia käyttäen. Seuraavissa kappaleissa käsitellään MS Accessin ominaisuuksia ja valintaan johtaneita tekijöitä.

MS Access on MS Office 2003 Pro -ohjelmistopakettiin kuuluva relaatiotietokannan hallintajärjestelmä (RDMBS), joka sisältää graafisen käyttöliittymän Microsoftin Joint Engine Technology (JET) -tietokantamoottorille sekä tietokantaohjelmistojen kehitysympäristön Access-tietokantasovellusten kehitykseen ja VBA-ohjelmointiin. MS Access on tarkoitettu pk-yrityksille, suurten yritysten organisaatioiden sisäiseen käyttöön sekä yksityisille henkilöille. Yrityksille MS Access pyrkii tarjoamaan kustannustehokkaan, nopeasti käyttöönotettavan ja joustavan alustan pienemmille tietokantaratkaisuille.

MS Access on itsenäinen tietokantasovellus joka mahdollistaa räätälöitävien ad hoc -tyyppisten työpöytäsovellusten toteuttamisen SQL-kyselyihin pohjautuvaan tietojenkäsittelyyn ja raportointiin. Access tukee henkilökohtaisia tietokantoja työasema-asennuksena. Accessia voidaan käyttää myös tietolähteenä ASP-kielisille websovelluksille, jotka toimivat Microsoft IIS-webpalvelimella. Access toimii myös tietokantana pienten käyttäjämäärien asiakas-palvelin sovelluksissa.

Tietolähteinä MS Access -asiakassovelluksissa voidaan käyttää toisten tiedostopohjaisten Access/Jet-lähteiden lisäksi laajempia palvelintietokantoja kuten Microsoft SQL Server -tuoteperheen tuotteita, tai muiden valmistajien kuten Oraclen tai MySQL:n tuotteita sekä mitä tahansa ODBC-yhteensopivaa tietolähdettä. MS Access tietokannan etuina MS Excelin tiedostopohjaiseen tietojenkäsittelyyn ovat esimerkiksi tietorakenteiden dynaamisuus, taulukoiden avaimiin, tietokenttien tyyppimääritteisiin ja taulukoiden välisiin suhteisiin perustuvat tiedontarkistukset eli rajoitteet. [5.]

Kuten muutkin MS Office -tuotteet MS Access tukee myös VBA:ta, joskin sen implementaatio on muista Office tuotteista poikkeava. Ensinnäkin, koska Accesissa makroilla kontrolloidaan kehitettävän tietokantasovelluksen toimintaa, eikä varsinaisesti Accessin käyttäytymistä. Accessissa ei myöskään ole Accessin omia objekteja sisällön käsittelyyn, kuten *Range*, *Worksheet* ja *Workbook*-objektit MS Excelissä, vaan tietojen käsittelyyn käytetään tietoyh-

teyskirjastoja ja sovelluskehitykseen VBA:n ohjelmakomponenttien objekteja.

Tietokantasovelluksen lomakkeiden kehitys tapahtuu Accessin omassa graafisessa kehittimessä ja Visual Basic Editoria käytetään lomakkeiden tapahtumakäsittelijöiden ohjelmointiin. Oman kehittimen etuna on Accessin tietokantatietoisten lomakkeiden ja komponenttien toteutus, joka mahdollistaa tietojen päivittämisen lomakkeille ja valintalistoille ilman ohjelmointia.

7.1 Access-sovelluksen hajauttaminen ja tietokantayhteydet

Seuraavat kappaleet käsittelevät tarkemmin MS Accessin soveltuvuutta hajautetun tietokantasovelluksen toteutukseen ja Access-sovelluksen tukea usean käyttäjän samanaikaiselle suoritukselle.

MS Accessissa on käytössä kaksi COM-objektikirjastoon kuuluvaa oliopohjaista tietoyhteysrajapintaa: ADO eli *Microsoft ActiveX Data Objects* ja DAO eli *Microsoft Data Access Objects library*.

Rajapinnat mahdollistavat tietokantatransaktioiden toteuttamisen VBA-sovelluksista tai mistä tahansa COM-arkkitehtuuria tukevasta ohjelmasta. Rajapinnan käyttö mahdollistaa myös Access-sovelluksen hajauttamisen eri tiedostoihin.

7.1.1 Data Access Objects library, DAO

DAO on vanhempi näistä kahdesta tietokantaobjektien käsittelyyn tuotetusta rajapinnasta VBA:n ja JET tietokantojen, ODBC-tietolähteiden sekä tiedostopohjaisten tietolähteiden välille. Sitä on kehitetty rinnan JET-tietokantamoottorin kanssa ja se tarjoaa ominaisuuksia ja menetelmiä, jotka on optimoitu JET-tietokantamoottorin ja MS Access-sovellusmoottorin väliseen toimintaan. DAO on poistunut virallisesti käytöstä, mutta se on edelleen suositeltu rajapinta sovelluksiin, jotka käyttävät pelkästään MS Access-tietokantoja. [16, 5, 4.]

MS Office 2000 ja 2002 -versioissa tietoyhteysmalleista oletuksena käytetään referenssiä ADO-malliin. MS Office 2003 -versio taas lisäsi DAO:n takaisin ja versio otti käyttöön molemmat referenssit oletuksena paremman alaspäin yhteensopivuuden tuottamiseksi. Tämä kuitenkin tuotti uuden ongelman, sillä molemmissa malleissa on samannimisiä objekteja.

Käytettäessä molempia referenssejä tulee ohjelmakoodissa käyttää täsmennettyjä objektimäärityksiä esiteltäessä tietyn mallin objekteja (esimerkiksi *DAO.Database, DAO.TableDef, DAO.Recordset, ADODB.Recordset*). Täsmennettyjen määritysten käytöstä on myös etua ohjelman suorituskyvylle, koska kääntäjän ei tarvitse tutkia jokaista kirjastohaaraa objektin määrittämiseksi. [15.]

7.1.2 Microsoft ActiveX Data Objects, ADO

ADO on yleisempi joukko objektimalleja, jotka toimivat rajapintana Jet-tietokantamoottoria käyttävien tietokantojen objektien lisäksi myös mihin tahansa ActiveX-tietokantamoottoria käyttävään tietokantaan, kuten Microsoft SQL Server-perheen tietokantoihin. Mallin yleiskäyttöisyysvaatimuksen johdosta kaikkia DAO:sta löytyviä erityisesti Jet-tietokantamoottoria varten suunniteltuja toimintoja ei löydy ADO:n toiminnoista. Malli mahdollistaa kuitenkin hajautetun tietokantasovelluksen kasvattamisen suuremmille MS SQL Server -tietokantapalvelimille sopivaksi hyvin pienellä vaivalla. [16.]

ADO-arkkitehtuuri sisältää kaksi päämallia:

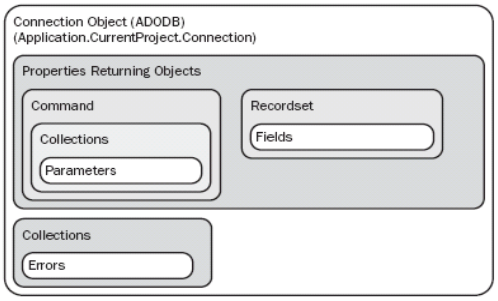
- ADODB on ActiveX Data Objects -perusmalli, jota käytetään tietuejoukkojen, toiminto- ja parametrikyselyiden käsittelyyn.
- ADOX, ADO Extensions for DDL and Security -mallia käytetään tietokantamoottorin katalogista löytyvien objektien käsittelyyn. Näitä ovat taulukot, näkymät, aliohjelmat ja määritellyt käyttäjät.

DAO yhdistyy eri tietolähteisiin ODBC-rajapinnan kautta. DAO:n kehittämisen jälkeen sovellusten arkkitehtuurivaatimukset kehittyivät itsenäisistä tietokantasovelluksista asiakas-palvelin-arkkitehtuurin sovelluksiksi. Todettiin, että tämä menetelmä ei tarjonnut tarpeeksi kattavaa ja monipuolista tukea erilaisille tietolähteille. [5.]

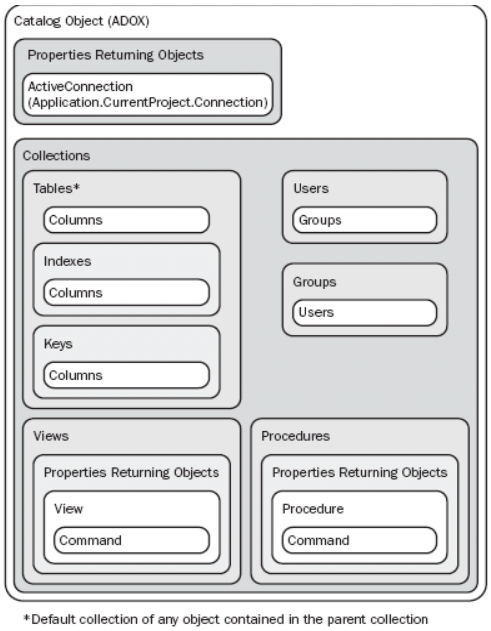
ADO-malli suunniteltiin juuri tätä silmälläpitäen, ja se on osa Microsoftin universaalin tietoyhteyden suunnitelmaa (UDA). ADO-malli toimii OLE DB (Object Linking and Embedding Databases) -rajapinnan päällä, joka hoitaa alemman tason transaktiot tietolähteiden kanssa. OLE DB tarjoaa laajemman tuen tietolähteille, ja mahdollistaa ADO:n kanssa käytettynä oliopohjaisen tietoyhteyden luomisen sovelluksesta ilman että sovelluksen tarvitsee tuntea tietolähteen toimintaa. VBA-sovelluksessa tarvitsee keskittyä vain tie-

toyhteyden (Connection) toimintaan ja sen objektien käsittelyyn. Kuitenkin tuki tietolähteen SQL-pohjaiselle käsittelylle on myös ADO:ssa saatavilla. [1]

Seuraavissa kaavioissa (kKuva 9 ja kKuva 10) on esitettynä ADODB:n sisältämät objektit (*Connection, Recordset, Command, Error*) ja pääominaisuudet sekä ADOX-mallin sisältämät objektit (Catalog, ActiveConnection, Tables, Views, Procedures, Users, Groups) ja pääominaisuudet, joita käytetään katalogista löytyvien objektien käsittelyyn. [1.]



Kuva 9: ActiveX Data Objects (ADODB) -mallin objektit



Kuva 10: ActiveX Data Objects Extensions for DDL and Security (ADOX) - mallin objektit.

7.1.3 Tietokantarajapintojen erot ja valintaan vaikuttavat tekijät

Tähän projektiin päätettiin ottaa käyttöön ADO-tietokantamalli koko sovellusarkkitehtuurissa. Pääasiassa päätökseen vaikutti ADO:n mahdollisuus laajentaa sovellus käyttämään muuta tietokantaa ja tämä mahdollistaisi jat-

kossa neuvottelut Process Guide Database -tietokantajärjestelmän käytöstä suoraan tietolähteenä sovellukselle.

Lähdemateriaalin perusteella listattiin seuraavat erot DAO- ja ADO-mallien välillä:

- DAO-malli tarjoaa natiivin tuen Access tietokannoille ja toimii tehokkaammin siinä ympäristössä kuin ADO. Access tarjoaa DAO-mallia käyttävälle sovellukselle myös tietolähteeseen liitetyn ehdottoman tietoyhteyden, jota Access ylläpitää. [1.]
- DAO-malli tukee parhaiten Jet-tietokannoille ominaista linkitettyjen taulujen mallia. Mikään muu tietoyhteysmalli ei tue tätä ominaisuutta natiivina. [1.]
- ADO-malli tarjoaa DAO:a monipuolisemman tuen ulkoisille tietolähteille ja suoran tuen vaihdettaessa tietokanta suurempaan MS SQL Server -tietokantapalvelimeen. [16.]
- ADO on jaettu useampaan kokoelmaan, jotta sovellukseen voidaan pakata vain tarpeelliset objektit, sovelluksen koko pienenee ja tehokkuus paranee. [5.]
- DAO mahdollistaa linkityksen useaan tietolähteeseen samanaikaisesti yhden objektin kautta. ADO:ssa taas yhteys luodaan aina yhteen tietolähteeseen, mutta yhteysobjekteja voidaan luoda useita rinnakkain. [16.]
- DAO mallin DBEngine-objektin kautta on mahdollista ajonaikaisesti lukea ja asettaa Jet-tietokantamoottorin ominaisuuksia. [1.]

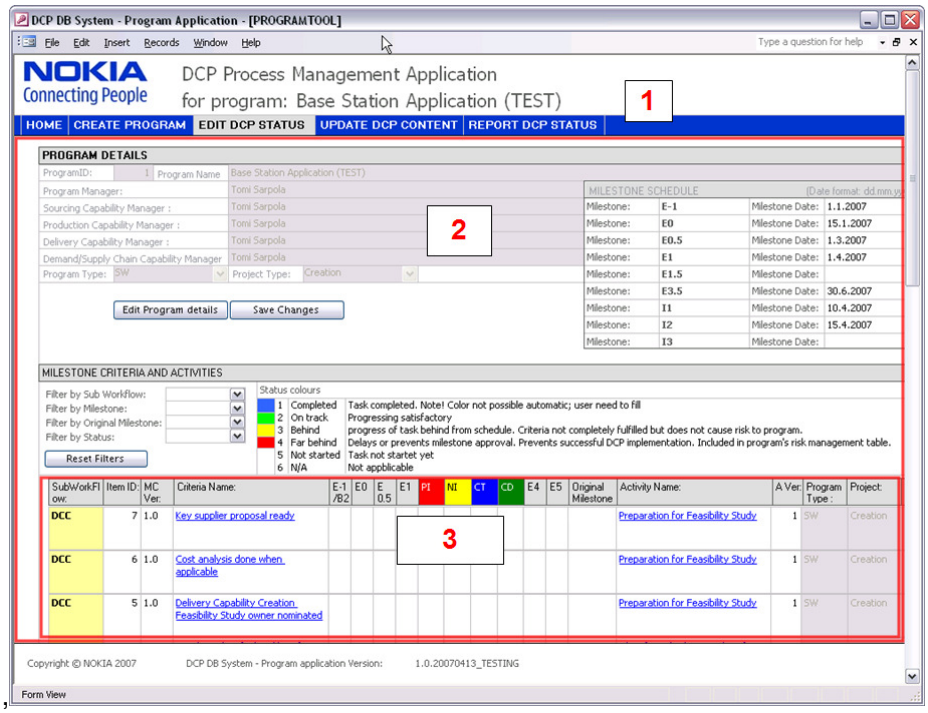
Oli myös tiedossa, että sovellus tullaan jakamaan keskustietokantaan ja asiakassovelluksiin, joten ADO soveltuu tähän hyvin. Kaikille ohjelmille sovellusarkkitehtuurissa päätettiin käyttää ADO-mallia, jotta ohjelman tietoyhteyksien käsittely olisi yhtenäistä ja voitaisiin käyttää mahdollisimman paljon samoja luokkia sekä keskustietokannan yhteyteen toteutettavan hallintasovelluksen että asiakassovelluksen kanssa.

7.2 VBA ohjelman hallittavuus, uudelleenkäytettävyys ja suunnittelumallit

Prosessinhallintajärjestelmän toteutuksen aikana tutkittiin MS Access -sovelluksen toteutukseen erilaisia menetelmiä, joiden tavoitteena oli parantaa ohjelmakoodin uudelleenkäytettävyyttä ja hallittavuutta.

7.2.1 Alilomakerakenne ja lomakkeiden uudelleenkäytettävyys

MS Accessin lomakkeet on yleensä sidottu tauluun tai usean taulun yhdistävään hakuun, josta lomakkeen tietokantatietoiset komponentit voivat hakea sisältöä. Tauluun sidottujen lomakkeiden lisäksi MS Access tukee sulautettuja alilomakkeita, jotka mahdollistavat monta-moneen-yhteyksiä käyttävien kyselyjen päivittämisen. [16.]



Kuva 11: MS Access alilomakkeiden käyttö Prosessinhallintajärjestelmässä

Prosessinhallintajärjestelmän (kKuva 11) hyödynnettiin alilomake-rakennetta taulujen tietojen linkittämisen lisäksi lomakkeiden uudelleenkäytettävyyden tuottamiseen. Koska jokaiselle toimintolomakkeelle tarvittiin toimintonapit sovelluksessa navigointiin, toteutettiin koko sovellukselle päälomake (kKuva 11 - kohta 1) ja siihen alilomake. Päälomake sisälsi toimintonappeja, joita painamalla vaihdetaan alilomakkeen esitystä (kKuva 11 - kohta 2).

Alilomakkeen esitykseen asetettiin myös toisia alilomakkeita (kKuva 11 - kohta 3), mikäli saman tiedon esittämiseen tarvittiin erilaisia näkymiä, kuten tässä etappivaatimusten näkymässä on tehty molemmille projektityypeille (HW ja SW), sillä kumpikin sisältää eri määrän etappeja.

Nämä alilomakkeet linkitettiin ylemmän tason lomakkeen kanssa sisältämään toisiinsa liittyvää tietoa päälomakkeella olevan alilomakekomponentin *SubForm.Link Child Fields* ja *SubForm.Link Master fields* -ominaisuuksien avulla (esim. kuvan lomakkeilla ProgramID-kentän perusteella).

Tämä kehys vähensi ohjelmointityötä tilanteissa, joissa yleisten osien komponenttisuunnitteluun tuli muutoksia (logo, otsikko, toimintonapit, versiotiedot) ja normaalisti muutokset olisi täytynyt kopioida jokaiselle lomakkeelle, jossa niitä käytettiin. Lisäksi kehys helpotti ohjelmiston hallintaa, koska oli mahdollista pilkkoa esityksiä loogisiin osiin ja näkymien räätälöintiä tarvittaessa voitiin keskittyä olennaisten näkymän osien räätälöintiin ja monistamiseen.

Osaltaan ohjelmointityötä vähensi MS Accessin dynaamisen listan eli jatkolomakkeen käyttö (kKuva 11 - kohta 3). Tämä rakenne yksinkertaistaa monta–moneen muotoisten suhteiden esityksen ohjelmointia, kun lomakkeen komponenttien ohjelmointi ja määrittely tehdään ainoastaan yhdelle datarivillle, jonka määrittelyt monistetaan dynaamisesti muille riveille. Jatkolomakkeiden ulkoasun parantamiseksi käytettiin myös ehdollisia muotoiluja, jotka tarkastavat kentän sisällön arvon ja muotoilevat kentän ulkoasun ehtojen perusteella. Sovelluksessa tätä ominaisuutta käytettiin etappivaatimuksen tilan esittämiseksi yleisesti käytössä olevilla tilavärityksillä.

7.2.2 Tapahtumakuuntelijat ja viestien välitys näkymissä

MS Access käyttää lomakkeiden tapahtumakäsittelijöissä Windowsin tapahtuma- ja viestipohjaista arkkitehtuuria. Tapahtuman kuten napin painalluksen yhteydessä Windows lähettää viestin sovellukselle ilmoittaakseen tapahtuman. Viesti sisältää yleensä kriittistä tietoa, jonka perusteella sovellus suorittaa tarvittavat toimenpiteet ja yleensä lähettää viestin takaisin Windowsille. Tämä menetelmä mahdollistaa interaktiiviset sovellukset, joissa käyttäjä vastaa sovelluksen tapahtumista. [1.]

MS Accessin lomakkeille tehtyjen toimintojen tapahtumat ohjataan Access-makroihin tai lomakekohtaisille VBA-proseduureille eli tapahtumakuuntelijoille jotka on määritetty lomakkeen *UserForm*-koodimoduulissa. VBA-tapahtumankuuntelijat mahdollistavat Access-makroja laajemman tuen sovelluksen automatisoinnille. Vaativampia toimintoja ovat esimerkiksi oletusarvojen

tuonti toisesta lomakkeesta, syöttörajoitteen muuttaminen toisen syötteen perusteella ja tiedottavien viestiruutujen luonti. [1, 16.]

Prosessinkäsittelysovelluksessa tärkeimpiä toimintoja, joissa tapahtumankäsittelijöitä tarvittiin, olivat edellisessä kappaleessa esitetyn kehyslomakkeen (kKuva 11 - kohta 1) painonapit ja niiden tuottamien toimenpiteiden toteutus sekä ponnahdusikkunoihin toteutetut dialogit. Painonappien tapahtumankäsittelijät asettavat päälomakkeella olevan alilomakkeen *SourceObject*-ominaisuudeksi tarvittavan sulautettavan lomakkeen nimen sekä asettavat alilomakkeen *Visible*-ominaisuuden tilaan *True*, jolloin sulautettavan lomakkeen ohjelmamoduuli suoritetaan ja asetetaan näkyväksi.

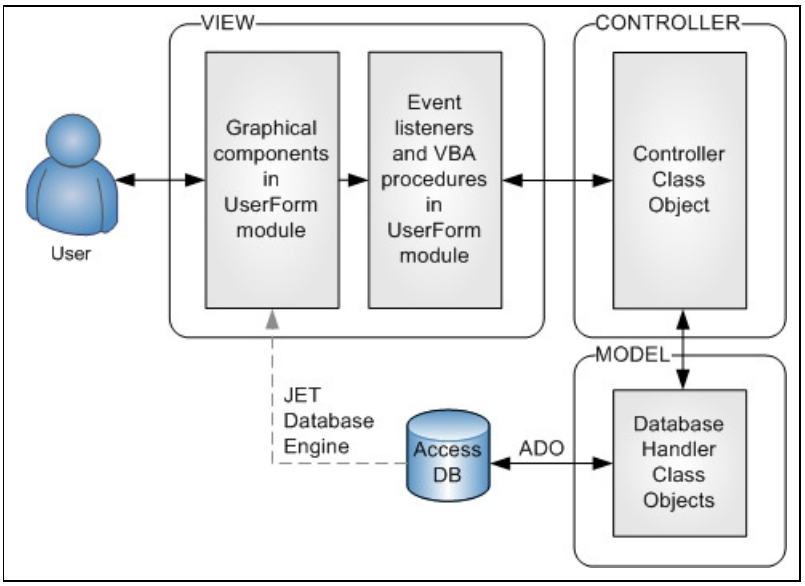
Ponnahdusikkunoissa käsiteltäviä dialogeja varten tarvittiin lomakeolioiden viestienvälitysmekanismi, jolla pystyttiin välittämään kaikki tiedot, joita ei tallennettu tietokantaan (esim. käsiteltävän tietorivin tunnisteen välitys dialogille sekä käyttäjän Cancel-napin painalluksella aiheuttama dialogin päättyminen). Muuttujien asettaminen avattavalle lomakkeelle ei ole suoraan mahdollista, koska avattavaa lomaketta ei ole vielä käännetty, jolloin lomakkeen muuttujia ei ole vielä olemassa. Avaamisen jälkeen taas muuttujien asettaminen ei onnistu, koska ohjelmakoodin suoritus ei palaa kutsuvaan lomakkeeseen ennen kuin dialogi on suljettu. [16.]

Viestinvälitysmekanismi toteutettiin *DoCmd*-metodin parametrien ja lomakeobjektien *Tag*-ominaisuuden avulla. *DoCmd*-metodin avulla lomake avataan ponnahdusikkunaan, ja metodin *OpenArgs*-parametrin sisässä voidaan lähettää *String*-tyyppistä dataa lomakkeelle. Tämä parametri on luettavissa lomakkeen *Form_Load*-tapahtuman yhteydessä lomakkeen omasta ominaisuudesta *Me.OpenArgs*.

Lomakkeen *Form_Close*-tapahtuman yhteydessä lomakkeen *Tag*-ominaisuuteen asetettiin jokin arvo, mikäli dialogissa tuotettiin tuloksia tai arvo -1, jos oli painettu Cancel-nappia ja dialogi lopetettiin. *Form_Close*-tapahtumassa lomake asetettiin ainoastaan piilotetuksi, jolloin ohjelmakoodin suoritus palasi lomaketta kutsuvaan ohjelmaan ja voitiin vielä lukea lomakkeen *Tag*-ominaisuuden arvo. Tämän jälkeen kutsuva ohjelma sulkee ja tuhoaa ponnahdusikkunan lomakkeen *DoCmd.Close*-metodilla.

7.2.3 MVC-mallin simulointi

Sovelluksen arkkitehtuurin suunnittelussa pyrittiin hyödyntämään opittuja suunnittelumalleja, jotta sovelluksesta saataisiin helpommin ylläpidettävä ja ohjelmakoodi jakautuisi loogisiin kokonaisuuksiin. Vaikka Visual Basic ja siitä periytyvä VBA eivät ole täydellisiä oliokieliä, tässä sovelluksessa tutkittiin, miten olioiden simulointia pystytään hyödyntämään kolmitasoisen MVC (Model-View-Controller) -mallin toteuttamiseksi (kKuva 12). MVC-malli jakaa sovelluksen kolmeen osajärjestelmään, jotka kommunikoivat keskenään.



Kuva 12: MVC-mallin mukainen sovellusarkkitehtuuri

MVC-mallissa käyttäjä suorittaa toimenpiteitä UserForm-moduulissa kuvatun lomakkeen graafisilla komponenteilla ja aktivoi tapahtumia. Jokaisella UserForm-moduulilla on komponentteja ja lomaketta vastaavat tapahtumankuuntelijat, jotka aktivoituvat käyttäjän toimenpiteistä ja lähettävät niistä tiedon sovelluksen kontrollerille, joka vastaa sovelluslogiikasta. Tapahtumasta riippuen kontrolleri välittää tiedon tietomallissa olevalle tietokantakäsittelijälle tai toiselle näkymälle.

Tietomalli vastaa tiedon käsittelystä kannassa ja kannan käyttöön liittyvistä proseduureista. Tietokantatoimenpiteiden jälkeen malli ilmoittaa kontrollerille toimenpiteen päättymisestä, jolloin kontrolleri voi käskeä näkymää päivittämään kenttänsä.

Näkymään voidaan välittää tietoa joko lähettämällä se kontrollerin kautta lomakkeen kutsun yhteydessä (esim. uuden käsiteltävän rivin tunniste, ID), lomake voi kysyä tietoja tietomallista suorittamalla kontrollerin metodeja, jotka puolestaan kutsuvat tietomallia. Näkymän tietokantatietoiset komponentit voivat lukea tiedon tietokantamoottorin kautta suoraan kannasta, jolloin kontrollerin täytyy käskeä lomaketta vain päivittämään itsensä kutsumalla alilomakkeen *Requery*-metodia.

Kontrolleri toteutettiin omana luokkaobjektina, johon ohjelmoitiin sovelluslogiikkaan liittyvät proseduurit. Koska Accessissa oletuksena luodaan ilmentymä sovelluksen käynnistävästä lomakkeesta ja luokkaobjekteja ei ole mahdollista asettaa käynnistymään ensimmäiseksi. Tässä sovelluksessa käynnistyväksi lomakkeeksi asetettiin *Main*-niminen *UserForm*-objekti ja kontrollerin ilmentymä asetettiin käynnistymisen yhteydessä luotavaksi *Main*-lomakkeen jäseneksi.

Tietomalliksi toteutettiin ADO-mallia käyttävä luokka, joka sisälsi tietokantatietueiden käsittelyyn liittyvän businesslogiikan. Kontrolleri sisälsi instanssit tietokantakäsittelijäluokasta ja kontrollerin kutsuessa sen metodeja tietokantakäsittelijä loi yhteyden sovelluksen tietokantaan ja suoritti vaaditut tietokantatoimenpiteet.

7.2.4 Luokkien uudelleenkäytettävyys

Itse luotuja VBA-luokkia voidaan myös siirtää sovelluksen ulkopuolelle omiksi tiedostoiksi. Tämä helpottaa sovellusten päivitystä, koska päivitykset voidaan tehdä pienempiin osuuksiin ja vaihtaa lennosta koskematta tietokannan sisältämään tiedostoon. Oletuksena luokkamoduuleilla on ilmentymän luomiseen vaikuttava *Instancing*-ominasuus asetettuna tilaan *Private*. Tämä tarkoittaa, että luokan ilmentymiä voi luoda vain kyseisessä projektissa. Toinen saatavilla oleva tila on *PublicNotCreatable*. Se taas tarkoittaa, että ulkoiset projektit voivat käyttää luokkaa, mutta eivät voi tehdä siitä ilmentymiä. Tästä syystä projektin, joka luokkaa tarjoaa, täytyy tarjota yleisessä makromoduulissa metodi luokan ilmentymän luomiseksi ja sen täytyy palauttaa luokan tyyppinen objekti ulkoiseen projektiin (Esimerkiksi luodaan omaa tietorakennetta varten *clsEmployee*-luokkamoduuli ja yleiseen makromoduuliin *New_clsEmployee*-funktio, *ClassProvider*-projektiin ja se samannimiseen tiedostoon) . [7, 1.]

Tämän jälkeen ulkoisessa projektissa olevat proseduurit voivat käyttää luokkaa, kuten mitä tahansa oletuskirjastoista löytyviä luokkia. Kuitenkin sillä erolla, että luokan ilmentymä saadaan ainoastaan luokkaa tarjoavan projektin *New_clsEmployee*-funktion avulla. Luokan alustuksessa käytetään, joko aikaista sidontaa (Early binding) tuomalla luokkaa tarjoava tiedosto muiden kirjastojen tapaan referenssinä projektiin, tai myöhäistä sidontaa (Late binding) viittaamalla tiedostoon suoraan ohjelmakoodista. Tässä on esitetty esimerkki molemmista tavoista:

```
Option Explicit
Sub UseExportedClass_EarlyBinding()
    Dim anEmployee As ClassProvider.clsEmployee
    Set anEmployee = ClassProvider.New_clsEmployee(Name:="Tomi
Sarpola")
    MsgBox anEmployee.Name
End Sub
Sub UseExportedClass_LateBinding()
    Dim anEmployee As Object
    Set anEmployee = Applications.Run("'g:\temp\class
provider.xls'!new_clsEmployee(Name:="Tomi Sarpola")"
    MsgBox anEmployee.Name
End Sub
```

Lisäksi MS Office 2003 Developer -versio sisältää Code Librarian -työkalun, joka on Visual Basic Editorin lisäosa. Sillä voidaan toteuttaa ja kääntää luokkia myös objektikirjastoiksi, joita voidaan käyttää vapaasti muissa projekteissa. [7.]

Tässä projektissa tutkittiin ulkoisten luokkien hyödyntämistä erityisesti sovellusten päivitykseen, mutta lopulta päädyttiin sulkemaan luodut luokat projektien sisään. Tämä yksinkertaistaa kuitenkin tuoteprojekteissa olevien loppukäyttäjien toimintaa, koska erillistä asentamista ei tarvita, vaan käyttäjä saa sekä tietokannan että sovelluksen käyttöönsä yhdestä tiedostosta.

7.2.5 VBA-lähdekoodin päivittäminen ohjelmallisesti

Projektissa tutkittiin myös sovelluksen hallintaa ja päivitysmahdollisuuksia VBA-lähdekoodin ohjelmallisen päivittämisen avulla. *Microsoft Visual Basic for Applications Extensibility 5.3* -objektikirjasto tuo käyttöön Visual Basic Editor -sovelluksen ja Office tiedostojen koodiprojektien VBA-koodimoduuleihin kohdistuvat objektit ja metodit:

```
Call Applications.VBE.VBProjects(str_FileName). _
  VBComponents(str_Class).CodeModule.InsertLines(i, str_line)
```

Jotta ohjelmallista lähdekoodin päivitystä voidaan käyttää loppukäyttäjän työasemalla, loppukäyttäjän täytyy muuttaa turva-asetusta, joka estää VBA-sovelluksen pääsyn ohjelmakoodiin. Asetus: ”*Trust access to Visual Basic Project*” löytyy MS Officen työkaluvalikoista kohdasta *Tools -> Macro -> Security -> Trusted publishers*. [7.]

Tällä menetelmällä sovelluksen päivitykset helpottuvat, kun ohjelmointi voidaan toteuttaa kehittäjän ympäristössä, ladata päivitetty ohjelmakoodi moduulitiedostoihin ja jakaa loppukäyttäjille päivityssovellus nk. huoltopaketti, jonka käyttäjät asentavat asiakassovellukseen.

Huoltopaketti ohjelmoidaan etsimään päivitystä vaativan asiakassovelluksen projektiobjekti Visual Basic Editorin tarjoamista tiedoista ja etsimään sen sisältämät VBA-ohjelmakoodit. Löydettyään ohjelmakoodit huoltopaketti poistaa vanhat ohjelmakoodit ja kirjoittamaan uudet huoltopakettiin linkitetyistä moduulitiedostoista. Ne taas voidaan säilyttää keskitetysti verkkopalvelimella, jolloin itse huoltopaketti-sovelluksen koko saadaan pieneksi.

Menetelmä ei ole kovin tehokas graafisten komponenttien ja komponenttisuunnittelun päivitykseen. Mutta mitä tahansa muutoksia tarvitaankin, ne voidaan myös toteuttaa ohjelmallisesti ja muokata komponenttien asetukset lomakkeen lataustapahtumassa suoritettavassa ohjelmakoodissa.

8 PROSESSINHALLINTAJÄRJESTELMÄ MS ACCESSILLA

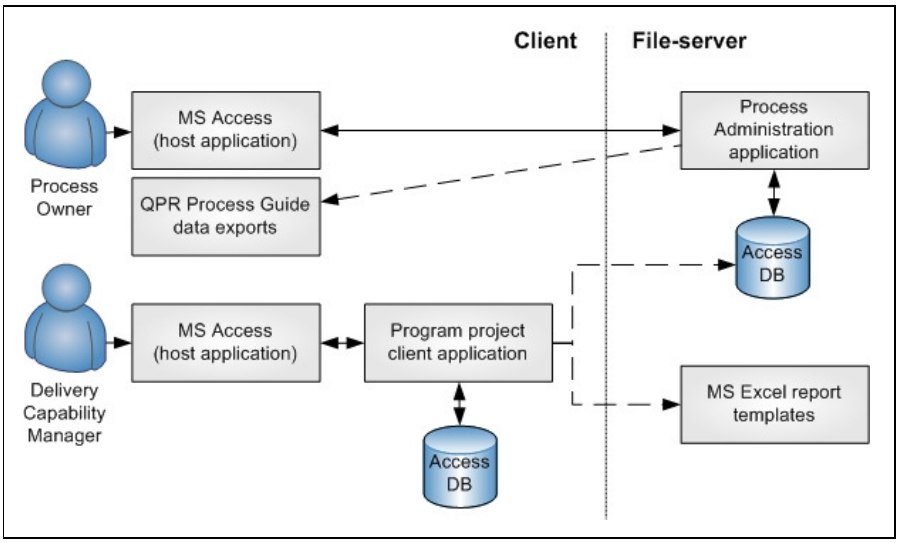
Prosessinkehitykseen liittyvän yritystoiminnan ja teknologiavaihtoehtojen tutkimuksen perusteella toimitusvalmiusprosessin kehitysryhmälle toteutettiin ja suunniteltiin prosessinhallintatyökalu, jonka tarkoituksena oli yksinkertaistaa prosessiin kuuluvan sähköisen materiaalin päivitystyötä, vähentää prosessipäivityksiin tarvittavaa työmäärää sekä parantaa sähköisen materiaalin käyttöä tuoteprojekteissa.

8.1 Prosessinhallintajärjestelmän yleiskuvaus ja arkkitehtuuri

Insinöörityössä toteutettavan ohjelmistoratkaisun toteutuksessa käytettiin hyväksi edellä kuvattuja tekniikoita sekä toteutusmalleja ja niihin perustuen

tuotettiin järjestelmä (kKuva 13), joka koostuu kahdesta VBA-sovelluksesta, kahdesta MS Access -tietokannasta sekä MS Excel -muotoisista raporttipohjista.

Järjestelmään tuotettiin prosessin kehittäjien käyttöön soveltuva Prosessinhallintasovellus, joka liitettiin suoraan keskustietokantatiedostoon. Prosessinhallintasovelluksesta oli mahdollista ottaa käyttöön QPR Process Guiden tietolistauksia.



Kuva 13: Prosessinhallintajärjestelmän arkkitehtuuri

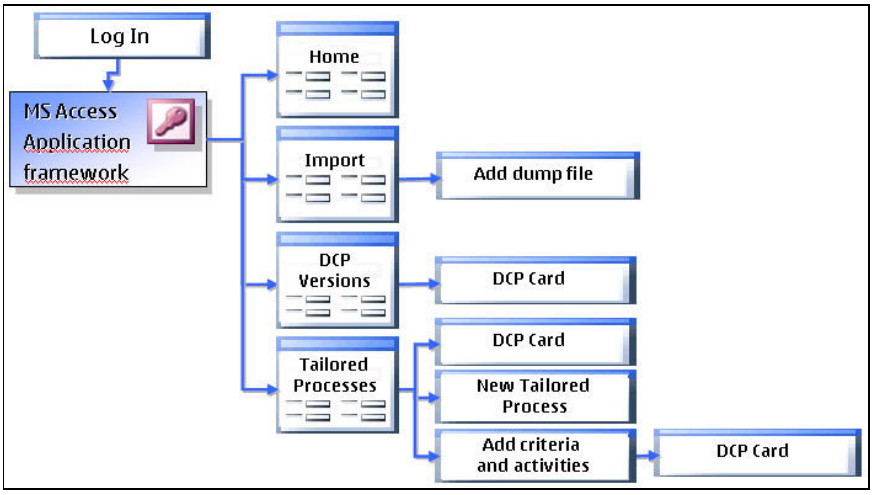
Keskustietokanta sisältää virallisen toimitusvalmiusprosessin ja sen versiot. Virallisten prosessien rinnalle tallennetaan räätälöityjä prosesseja, joita voidaan käyttää mallina esimerkiksi tietyn tuoteperheen tuoteprojekteissa.

Tuoteprojekteissa toimivat toimitusvalmiusprojektin johtajat käyttävät prosessia Tuoteprojektin asiakassovelluksesta, joka tallennetaan käyttäjän työasemalle. Tuoteprojektin asiakassovellus pystyy ottamaan yhteyden keskustietokantaan prosessien sisällön päivitystä varten. Lisäksi asiakassovellus kykenee tuottamaan seurantaraportteja tuoteprojektin toimitusvalmiusprosessista.

8.2 Prosessinhallintasovellus ja keskustietokanta

Toimituskykyprosessin kehitysryhmän jäsen käyttää järjestelmän Prosessinhallintasovellusta ylläpitääkseen QPR Process Guide -järjestelmästä ladattuja prosessitietoja. Alla olevassa kuvassa (kKuva 14) on kuvattu sovelluksen

näkymät ja niiden välillä tapahtuvat siirtymät. Suuremmat kuviot kuvassa ilmaisevat sovelluksen kehyslomakkeelle (kKuva 11) avattavia pääikkunoita. Pienemmät kuviot taas ilmaisevat ponnahdusikkunoihin avautuvia dialogeja.



Kuva 14: Prosessinhallintasovelluksen näkymät ja siirtymät

Prosessinhallintasovellus sijaitsee tiedostopalvelimella fyysisesti samassa tiedostossa, kuin järjestelmän keskustietokanta, koska sovelluksella hallitaan virallisten prosessikuvausten tietueita.

8.2.1 Prosessiversioiden luonti QPR Process Guide -tietolistauksista

Sovelluksen käynnistyessä MS Access -ikkunassa käyttäjälle esitetään ensin kirjautumislomake, johon käyttäjä asettaa tietokantaan kirjatun prosessin kehittäjän tunnuksen ja salasanan. Tämän jälkeen sovelluksen kehyslomake avataan ja siihen ladataan oletusnäkymänä *Home*-lomake, joka sisältää kuvaukset muiden pääikkunoiden toiminnoista ja tarkoituksesta.

Ladatakseen uuden version prosessista käyttäjä siirtyy *Import*-lomakkeelle, missä käyttäjältä pyydetään tiedot uuden version kirjaamista varten sekä uuteen versioon tuotavien tietojen sijainnit. QPR Process Guide -tietolistausten tiedostot haetaan *Add dump file* -dialogin avulla. Dialogissa käytetään standardia Windowsin tiedostonavaus-toimintoa tiedostojen paikantamiseen. Valittuihin tiedostoihin linkitetään tiedot prosessin osasta ja sisällön tyypistä.

Kun kaikki tiedostot on linkitetty Import-lomakkeelle, käyttäjä aktivoi VBA-ohjelman, joka lukee jokaisen tiedoston toiseen kahdesta käytettävästä väliaikaistaulusta. Taulut on varattu erikseen etappivaatimuksille ja prosessinakti-

viteettikuvauksille. Tiedostojen lukemisen jälkeen ohjelma linkittää väliaikaistauluissa olevat etappivaatimukset ja aktiviteettikuvaukset yhteen prosessialkio-tietueeksi. Kullekin alkiolle tehdään tarkistus, onko vastaavaa paria jo kuvattuna kantaan. Jos tietuetta tai sen versiota ei löydy ohjelma luo kantaan joko uuden prosessialkio-tietueen tai uuden version olemassa olevalle prosessialkiolle. Etappivaatimusten ja aktiviteettikuvausten linkittämiseen käytetään QPR Process Guiden tietokantaan kirjattavaa vaatimuksen nimeä. VBA-ohjelma tarkastaa tämän manuaalisen syötteen virheiden varalta ja ilmoittaa käyttäjälle ohjelman suorituksesta kerätyssä lokissa virheilmoituksen, mikäli jossakin tiedostossa on virheellinen rivi. Kun prosessiversion tuonti on valmis, käyttäjälle esitetään automaattisesti *DCP Versions* -päälomake.

8.2.2 Prosessiversioiden näkymä ja niiden sisällön tarkastelu

Viralliset toimitusvalmiusprosessin versiot listataan *DCP Versions* -lomakkeelle listana, josta käyttäjä voi valita yhden prosessiversion kerrallaan tarkasteltavaksi. Valinnan jälkeen prosessin nimi ja kuvaus tuodaan muokkauslomakkeelle, jossa käyttäjä voi tehdä niihin muutoksia. Käyttäjä voi myös tarpeen vaatiessa tuhota valitun version. Valitulle versiolle esitetään jatkolomake-tyyppistä komponenttia käyttäen myös kaikki prosessiversioon liitetyt etappivaatimukset sekä niille linkitetyt prosessiaktiviteetit listana. Jatkolomakkeen rivillä oleviin komponentteihin on liitetty koko rivin kattava tapahtumakuuntelija, joka avaa *DCP Card* -dialogin ponnahdusikkunassa.

DCP Card -lomakkeella käyttäjälle esitetään kaikki tiedot valitusta etappivaatimuksesta ja siihen liittyvästä prosessiaktiviteetistä. Lomakkeen kautta käyttäjä voi editoida myös etappivaatimuksen tai aktiviteetin tietoja. Näiden tietojen lähde on kuitenkin QPR Process Guide -järjestelmän tietokannassa, joten muutoksia ei ole yleensä tarvetta tehdä.

8.2.3 Räätälöityjen prosessien näkymä ja niiden sisällön tarkastelu

Vastaavasti virallisista prosesseista muokatut räätälöidyt prosessit, jotka ovat kirjattuna kantaan, esitetään käyttäjälle *Tailored processes* -päälomakkeella. Niille on myös listanäkymä, joka esittää eri räätälöidyt prosessit ja listan valinnasta prosessi tuodaan muokkauslomakkeelle sekä prosessin etappivaatimukset ja aktiviteetit esitetään jatkolomakkeella listattuna.

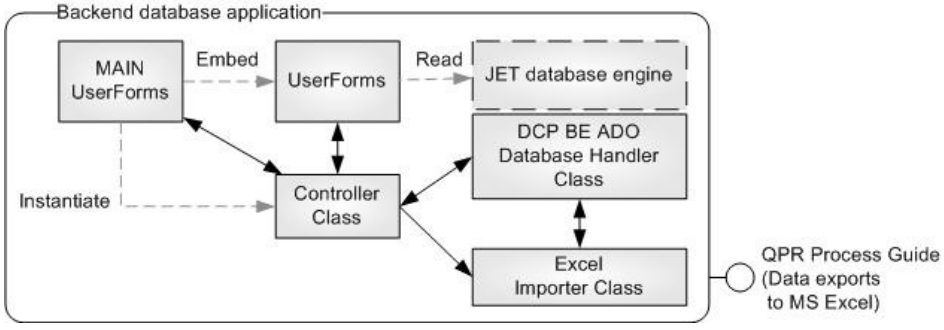
Räätälöityjen prosessien listan toimintoihin kuuluu sekä olemassa olevien tietueiden tuhoaminen, että uusien luominen. Uuden rivin luomista varten avataan *New Tailored Process* -lomake, jossa räätälöidylle prosessille annetaan kuvaus ja prosessiin liittyviä määritteitä. Lomakkeella esitetään myös viimeisin virallinen prosessiversio, josta räätälöityyn prosessiin tuodaan lisäyksen yhteydessä etappivaatimukset ja aktiviteettikuvaukset.

Päälomakkeen etappivaatimusten ja aktiviteettien listan riviä painettaessa käyttäjä saa esiin *DCP Card* -lomakkeen ponnahdusikkunassa, missä käyttäjä voi muokata aikataulua, johon etappivaatimus on sidottu, tai kokonaan poistaa etappivaatimuksen.

Kun räätälöityyn prosessiin halutaan poistamisten jälkeen lisätä etappivaatimuksia tai aktiviteettikuvauksia, käyttäjä painaa päälomakkeen lisäys nappia, jonka jälkeen avataan *Add criteria and activities* -lomake ponnahdusikkunassa. Lomake sisältää listan viimeisimmän virallisen prosessin etappivaatimuksista ja aktiviteeteista. Listan riviä painamalla käyttäjä saa jälleen *DCP Card* -lomakkeen, jolta käyttäjä voi tarkastaa lisättävän tietueen tiedot ja suorittaa lisäyksen.

8.2.4 Sovellusarkkitehtuuri

Seuraava kuva esittää Prosessinhallintasovelluksen VBA-ohjelman sisäisen rakenteen. Sovellus rakennettiin aiemmin kuvatun MVC-mallin mukaisesti siten, että sovelluslogiikasta vastaa *Controller*-luokka, josta muodostetaan yksi ilmentymä sovelluksen käynnistyksen yhteydessä käännettävän *Main*-nimisen lomakkeen jäseneksi. Tämä lomake toimii sovelluksen kehyksenä ja ainoastaan tarjoaa erilaisia näkymiä, joita kontrollerissa sille käsketään näytettäväksi.



Kuva 15: Prosessinhallintasovelluksen luokkakaavio

Tietokannan käsittelystä vastaa *DCP BE ADO Database Handler* -luokka, joka muodostaa tässä sovelluksessa paikallisia yhteyksiä ADO-rajapinnan kautta samassa tiedostossa olevaan keskuskantaan. Tätä luokkaa voidaan suoraan hyödyntää seuraavissa sovellukseen liittyvissä projekteissa, koska se sisältää kyseiseen tietokantaan käytettävän yritystoiminnan logiikan ja tarjoaa rajapinnat tietojen hakua ja päivitystä varten VBA:n proseduureina. Lisäksi lomakkeiden eli *UserForms*-objektien tietokantatietoiset komponentit kykenevät lukemaan JET-tietokantamoottorin avulla sisältöä kannasta.

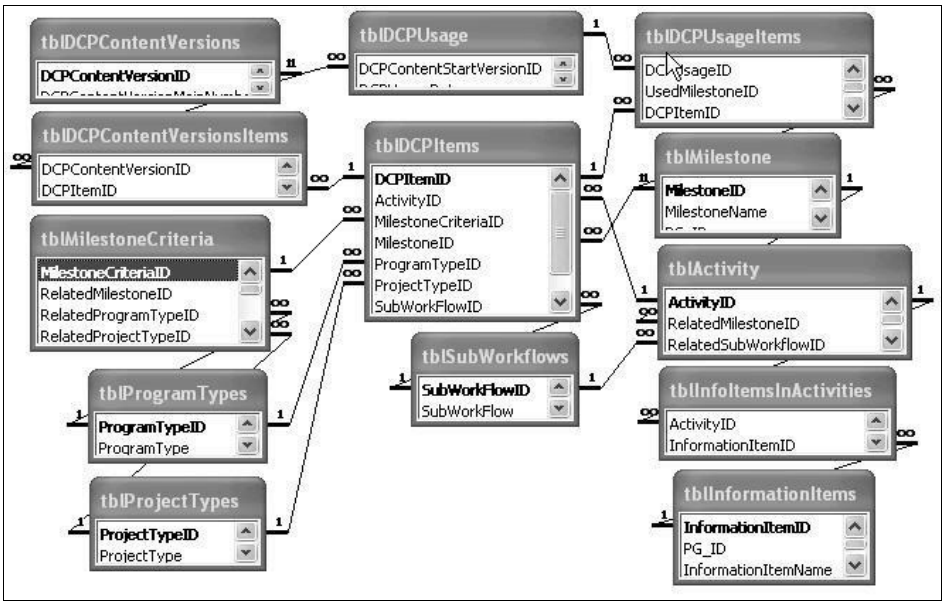
Kuten sovelluksen toiminnan kuvauksesta kävi ilmi, sovelluksessa oli useita samantyyppisiä operaatioita toteutettuna eri sovelluksen osiin ja näiden suunnittelussa pyrittiin hyödyntämään mahdollisimman näkymien ja lomakkeiden uudelleenkäytettävyyttä. *Main*-kehyslomake kattoi yhdellä *UserForm*-moduulilla kaikkien päälomakkeiden navigointiin liittyvät toiminnot. *DCP Card* -lomaketta käytettiin myös useassa eri tilanteessa ja siihen lisättiin tarpeen mukaan alilomakkeena tilanteen vaatimia toimintonappeja.

QPR Process Guiden tietolistaukset luetaan sovelluksessa *Excel Importer* -luokan toimintojen avulla, joka kykenee käsittelemään Excel-formaatin tiedostoja ja lukemaan niistä prosessisisältöä sovitun muotoisina taulukoina. Luokka kysyy *DCP BE ADO Database Handler* -luokalta käytettävät tiedostot, luo väliaikaistaulukot ja välittää sen jälkeen *DCP BE ADO Database Handler* -luokalle prosessialkiot, jotka tiedostosta ladattiin.

8.2.5 Tietokanta

Keskustietokannan suunnittelussa otettiin lähtökohdaksi etappivaatimusten ja prosessiaktiviteettien tietojen yhteen liittäminen ja prosessin sisällön versiointi päivitysten yhteydessä. Tietokannan 13 taulua sisältävä normalisoitu rakenne on esitettynä kuvassa Kuva 16.

Etappivaatimusten ja prosessiaktiviteettien linkittäminen prosessiversioihin toi kantaan useita monta-moneen-suhteita, joten kaikki kannan taulut ovat indeksoitu pääavaimella, josta muodostettiin vierasavaimia suhteeseen liittyviin tauluihin. Pääavaimena kaikissa tauluissa on käytetty automaattisesti lukuarvoaan nostavaa kokonaislukukenttää.



Kuva 16: Prosessinhallintasovelluksen tietokannan relaatiokaavio

Kannassa *tblActivity*-tauluun kirjattavat prosessiaktiviteetit liitetään *tblMilestoneCriteria*-taulun etappivaatimuksiin käyttämällä välitaulua *tblDCPItems*, missä nämä kaksi tietoa muodostavat nk. prosessialkion. Kukin prosessialkio on liitetty yhteen etappiin *tblMilestone*-taulussa ja yhteen aliprosessiin *tblSubWorkflows*-taulussa, nämä tiedot määräytyvät prosessiaktiviteetin perusteella. Lisäksi alkio on liitetty yhteen tuoteprojektityyppiin *tblProgramTypes*-taulussa ja yhteen projekti vaiheeseen *tblProjectTypes*-taulussa, nämä määräytyvät etappivaatimuksen perusteella.

Yksi alkio voi liittyä mihin tahansa tblContentVersions-taulun prosessiversioista ja vastaavasti prosessiversio voi sisältää minkä tahansa alkion, joten näiden taulujen linkittämiseen käytetään tblContentVersionsItems-taulua.

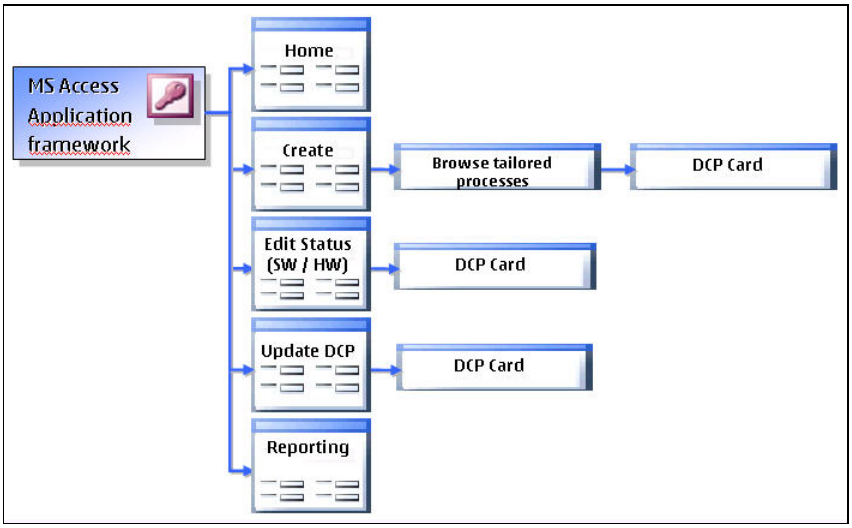
Vastaavasti räätälöityjä prosesseja varten on taulut *tblDCPUsage* ja *tblDCPUsageItems*. Lisäksi yksi räätälöity prosessi perustuu aina johonkin viralliseen prosessiin ja *tblDCPUsage*-taulussa käytetään vierasavaimena virallisen prosessin pääavainta.

8.3 Tuoteprojektin asiakassovellus ja projektin tietokanta

Tuoteprojekteissa toimivat toimitusvalmiusprojektin johtajat tekevät toimitusvalmiusprojektin, jossa otetaan käyttöön toimitusvalmiusprosessi ja seurataan sen edistymistä tuoteprojektin etappivaatimusten aikataulussa. Proses-

sin käyttöönotto ja seuranta tehdään Tuoteprojektin asiakassovelluksesta. Asiakassovellus jaetaan käyttäjille Access-tiedostona, jonka käyttäjä tallentaa tuoteprojektin dokumentaatioon tai omalle työasemalle.

Alla olevassa kuvassa on kuvattu sovelluksen näkymät ja niiden välillä tapahtuvat siirtymät. Suuremmat kuviot kuvassa ilmaisevat sovelluksen kehyslomakkeelle (kKuva 11) avattavia sisältöikkunoita, pienemmät kuviot taas ilmaisevat ponnahdusikkunoihin avautuvia dialogeja.



Kuva 17: Tuoteprojektin asiakassovelluksen näkymät ja siirtymät

Asiakassovelluksen tiedostossa on projektikohtaiseen prosessinseurantaan tietokanta, johon tallennetaan tuoteprojektin tiedot, projektin etappien aikataulu sekä tuoteprojektille sovittu toimitusvalmiusprosessi. Kehitysprojektissa katsottiin, että tuoteprojektien prosessien seurantaan ei olisi kannattavaa tuottaa keskitettyä tietokantaa, vaan prosessinseurantakanta tehdään jokaiselle tuoteprojektille. Tähän johti vaatimukset, että tuoteprojektikohtainen seuranta tulisi olla mahdollista ilman verkkoyhteyttä ja että sovelluksen käyttöönoton ei tulisi vaikeuttaa olemassa olevan tiedostopohjaisen prosessin seurantamallin toimintaa.

8.3.1 Tuoteprojektin perustaminen asiakassovelluksen tietokantaan

Sovelluksen käynnistyessä MS Access -ikkunassa käyttäjälle esitetään oletuksena sovelluksen kehyslomake. Siihen ladataan oletusnäkymänä *Home*-lomake, joka sisältää kuvaukset muiden pääikkunoiden toiminnoista ja tarkoituksesta.

Perustaakseen tuoteprojektin tiedot asiakassovelluksen tietokantaan käyttäjä siirtyy *Create*-lomakkeelle. Tässä vaiheessa sovellus lataa viimeisimmät prosessitiedot ja saatavilla olevat räätälöidyt prosessit tiedostopalvelimella olevasta keskustietokannasta. *Create*-lomakkeella käyttäjältä pyydetään tiedot tuoteprojektin tyypistä ja käyttäjä voi valita mistä toimitusvalmiusprosessin sisältö ladataan: joko viimeisimmästä virallisesta prosessista tai valitusta räätälöidystä prosessista.

Kun käyttäjä valitsee räätälöidyn prosessin lähteeksi, käyttäjälle esitetään *Browse tailored processes* -lomake ponnahdusikkunassa. Siinä esitetään kaikki räätälöidyt prosessit listana sekä lisäksi lista räätälöityyn prosessiin kuuluvista etappivaatimuksista ja aktiviteeteista. Nämä taas esitetään listan painalluksesta *DCP Card* -lomakkeella ponnahdusikkunassa, mistä käyttäjä voi katselmoida etappivaatimuksen tai aktiviteetin sisältöä. Kun tuoteprojektin perustaminen on suoritettu, käyttäjälle esitetään automaattisesti *Edit Status* -päälomake.

8.3.2 Tuoteprojektin toimitusvalmiusprosessin seuranta

Edit Status -päälomakkeella käyttäjä voi muokata tuoteprojektin tietoja, projektin etappien aikataulua sekä seurata etappivaatimusten ja prosessiaktiviteettien tilaa. Sovellukseen toteutettiin kaksi eri näkymää, jossa tuoteprojektin toimitusvalmiusprosessin seurantaa voitiin hoitaa. Näkymät toteutettiin kahdelle eri tuotetyypille. Molempien projektien etappiaikataulun esitys ei ollut mahdollista yhden tyyppisellä lomakkeella, koska molemmissa oli erilaisia etappeja. HW-projektiin eli tuotetta valmistavaan projektiin kuului valmistusvaiheen etapit E -1:destä E5:een, ylläpitovaiheen etapit E5:destä E6:een sekä alasajovaiheen etapit E6:desta E10:een. SW-projektiin eli ohjelmistoa valmistavaan projektiin kuului valmistusvaiheen etapit E -1:destä E5:een sekä E3:n yhteydessä olevat iteratiivisten vaiheiden etapit.

Käyttäjän avatessa *Edit Status* -päälomaketta sovellus tarkastaa kantaan luodun tuoteprojektin *Program type* -kentästä kumpaa projektityyppiä (HW tai SW) käytetään ja esittää sen perusteella päälomakkeen alilomakkeessa projektityypille sovitetun etappitaulukon, jossa etappivaatimukset ja prosessiaktiviteetit listataan. Käyttäjä tutkii etappivaatimusten ja prosessiaktiviteettien listassa olevien rivien tilaa selaamalla taulukkoa. Sovellukseen tuotettiin selauksen helpottamiseksi suodatinnappeja, joilla listasta voidaan poimia ha-

luttuja rivejä suodattimen valinnan mukaan (esim. tiettyyn etappiin kuuluvat etappivaatimukset).

Etappivaatimusten ja prosessiaktiviteettien tila esitetään etappitaulukossa graafisesi yleisesti käytössä olevan viisivärisen tilaesityksen mukaisesti, joka toteutettiin jatkolomakkeelle ehdollisten muotoilujen avulla. Tilaa voidaan muuttaa etappitaulukon kenttää painamalla tai siirtymällä *DCP Card* -lomakkeelle, joka avataan ponnahdusikkunaan, kun käyttäjä painaa etappivaatimuksen tai aktiviteetin nimeä.

DCP Card -lomakkeella käyttäjä voi tilan muuttamisen lisäksi siirtää prosessiaktiviteetin ja etappivaatimuksen toteutettavaksi toiseen etappiin, kirjoittaa tilaa koskevia huomioita ja muistiinpanoja raportointia varten sekä kirjata aktiviteettiin ja vaatimukseen liittyvää palautetta, joka välitetään sovelluksesta prosessinkehittäjille.

8.3.3 Asiakassovelluksen tietokannan päivittäminen

Mikäli tuoteprojektissa käytettävään prosessiin tarvitaan muutoksia, joko lisäyksiä esimerkiksi päivittyneestä virallisesta projektista, tai tuoteprojektin prosessista täytyy poistaa aktiviteetteja. Tämä onnistuu sovelluksen *Update DCP* -päälomakkeella.

Kun käyttäjä avaa *Update DCP* -lomakkeen, sovellus ottaa yhteyden keskuskantaan ja päivittää tuoteprojektin perustamisen yhteydessä ladatun prosessidatan. Päivityksen jälkeen lomake avataan, ja käyttäjälle esitetään lista viimeisimmän virallisen prosessin etappivaatimuksista ja aktiviteeteistä, josta näitä voi lisätä tuoteprojektiin. Lisäksi esitetään lista tuoteprojektissa jo tällä hetkellä olevista etappivaatimuksista ja aktiviteeteistä näiden poistamista varten. Varsinainen lisäys ja poisto tehdään listojen rivien painalluksesta avautuvalla *DCP Card* -lomakkeella.

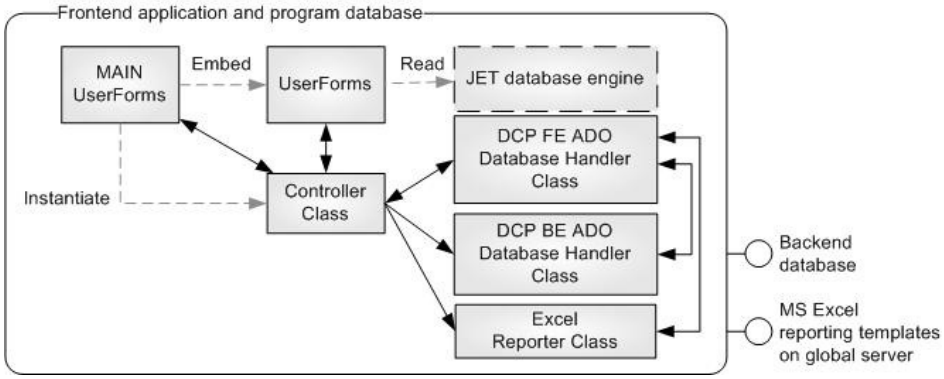
8.3.4 Tuoteprojektin toimitusvalmiusprosessin raportointi

Tuoteprojektikohtainen prosessin seuranta oli mahdollista raportoida MS Excel-raporteille *Reporting*-päälomakkeen näkymästä. Raportointia varten tuotettiin kolme raporttipohja*a: Activity listing report*, *Milestone criteria status report* ja *Lessons Learned feedback report*. Raportointitoiminto aktivoitiin kunkin raporttityypin toimintonapista, jolloin sovellus keräsi tuoteprojektin tietokannasta etappivaatimusten ja aktiviteettien tiedot, avasi tiedostopalvelimella

olevan valmiiksi muotoillun raporttipohjan, ja tulosti keräämänsä tiedot raportiksi. *Reporting*-päälomakkeen toimintoihin kuului myös tiedonsiirtotoiminto, jolla projektille räätälöity prosessi voitiin julkaista keskustietokantaan ja sitä voitiin sen jälkeen käyttää pohjana perustettaessa uusia tuoteprojekteja toiseen asiakassovelluksen tietokantaan.

8.3.5 Sovellusarkkitehtuuri

Kuva 18 esittää Tuoteprojektin asiakassovelluksen VBA-ohjelman sisäisen rakenteen. Sovellus rakennettiin Prosessinhallintasovelluksen pohjalta ja noudattaa samaa MVC-mallia.



Kuva 18: Tuoteprojektin asiakassovelluksen luokkakaavio

Asiakassovelluksessa paikallisen tietokannan tiedoista vastaa siihen sovitettu *DCP FE ADO Database Handler* -luokka. Keskustietokannan yhteyksiä varten kopioitiin Prosessinhallintasovelluksesta *DCP BE ADO Database Handler* -luokka hallinnoimaan keskustietokannan toimintoja. Luokkaan tarvittiin vaihtoehtoinen yhteysmääritys asiakassovelluksen yhteyksiä varten.

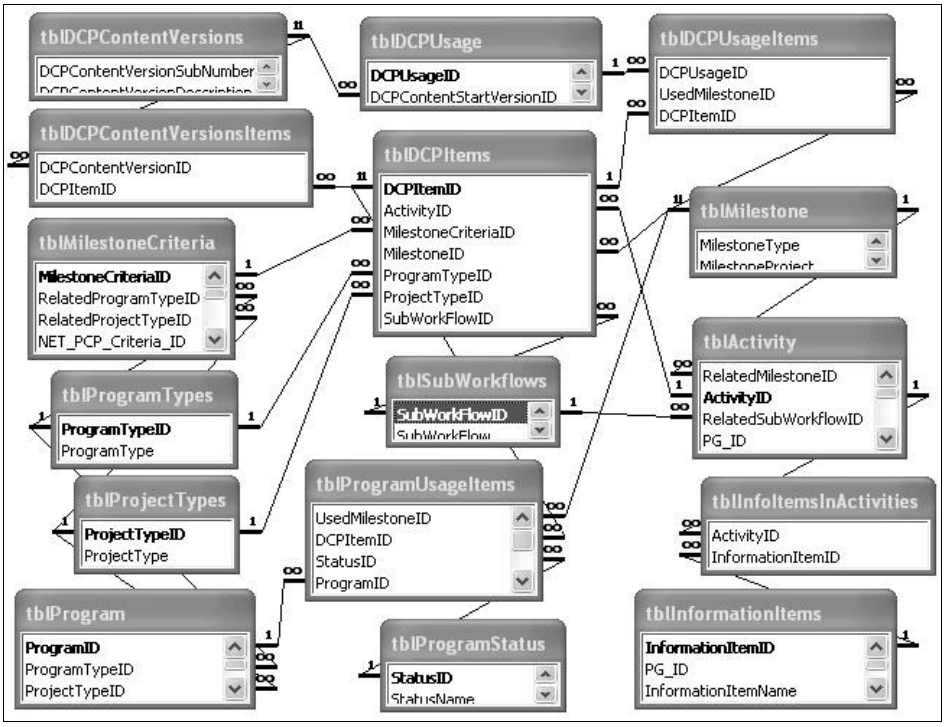
Tuoteprojektin asiakassovellus käyttää *Excel Reporter* -luokkaa tiedostopalvelimella olevien raporttipohjien käsittelyyn. Luokka kysyy *DCP BE ADO Database Handler* -luokalta tiedostojen sijainnit sekä kokoelman prosessialkioita, jotka täytetään raporttitiedostojen kenttiin.

8.3.6 Tietokanta

Tuoteprojektin asiakassovelluksen tietokannan pohjana käytettiin suoraan keskustietokannan rakennetta, koska prosessisisältö ladataan samassa muodossa asiakassovellukseen. Lisäksi tarvittiin taulut tuoteprojektin tietojen, etappiaikataulun ja projektin toimitusvalmiusprosessin tilan tallennusta

varten. Tietokanta (kKuva 19) sisälsi keskuskannan 13 taulun lisäksi kolme taulua asiakassovelluksen tietoja varten.

Pääavaimena kaikissa prosessisisältöä koskevissa tauluissa käytetään keskuskannasta saatavia indeksejä, joten ne kentät tallennettiin tavallisina kokonaislukukenttinä. Tuoteprojektitietojen tauluissa pääavaimina käytetään automaattisesti lukuarvoaan nostavaa kokonaislukukenttää.



Kuva 19: Tuoteprojektin asiakassovelluksen tietokannan relaatiokaavio

Taulu *tblProgram* sisältää asiakassovellukseen perustetun tuoteprojektin tiedot. Projekti on liitetty aina yhteen tuoteprojektityyppiin *tblProgramTypes*-taulussa ja yhteen projekti vaiheeseen *tblProjectTypes*-taulussa. Näiden perusteella määräytyy, millaisia *tblDCPItems*-taulun sisältämiä nk. prosessialkioita projekti voi käyttää.

Tuoteprojektille käytettävät prosessialkiot listataan *tblProgramUsageItems*-taulussa, mihin kirjataan alkion pääavain. Lisäksi tallennetaan sen etapin pääavain, johon projekti on päättänyt alkion toteutettavaksi. Prosessialkio liitetään myös yhteen tilaan *tblProgramStatus*-taulussa, mikä ilmaisee etappivaatimuksen ja prosessiaktiviteetin suorituksen vaihetta.

8.4 Testaus ja käyttöönotto

Prosessinhallintajärjestelmän testaus toteutettiin kehityksen aikana moduuli ja integraatiotestauksena ja lisäksi projektiryhmän avustuksella suoritettiin järjestelmätestausta projektin aikataulun mukaisesti aina loogisen kokonaisuuden valmistuttua. Järjestelmätestaus suoritettiin käyttäjärooleihin perustuvien käyttötapausten avulla, jolloin sovelluksen yhteensopivuus yritystoiminnan menetelmiin ja loppukäyttäjien vaatimuksiin tuli intuitiivisesti testattua. Käyttötapauksiin kirjattiin myös systemaattisen testauksen toimintalistauksia niille sovelluksen alueille, jotka käsittelivät tietojen päivitystä.

Sovelluksen testauksessa havaittiin MS Accessin käyttöliittymän toimintaan liittyviä ongelmia sekä joitain Excel-tiedostojen käsittelyyn liittyviä ohjelmavirheitä. Näiden korjaaminen kuitenkin viivästytti alkuperäistä projektisuunnitelmaa. Järjestelmä saatettiin käyttöönottokuntoon maaliskuussa 2007.

8.5 Prosessinhallintajärjestelmän kehitysideat

Järjestelmä saatiin toteutettua vastaamaan projektin alussa asetettuja vaatimuksia, mutta projektin testauksen perusteella järjestelmälle kirjattiin joitain kehitysideoita, joita projektin resurssien puitteissa ei ollut mahdollista toteuttaa:

- Prosessinhallintasovelluksen prosessiversion luontitoiminnossa olevaa Excel-tiedostoja käsittelevää VBA-ohjelmaa tulisi parantaa niin, että se ilmoittaisi yksittäisen tiedostojen solun lokissa, mikäli tietojen linkitykseen käytettävässä manuaalisessa syötteessä havaitaan virhe. Tällä hetkellä ohjelma kykenee ilmoittamaan vain tiedoston, koska kaikki tiedot ladataan väliaikaistauluun yhdellä operaatiolla.
- Tuoteprojektin asiakassovelluksen graafisia toimintoja ja syötemahdollisuuksia tulisi parantaa niin, että sovelluksesta tulisi joustavampi ja että se soveltuisi paremmin projektinhallinnan toimenpiteisiin. Tämä vaatisi graafisten kalenteritoimintojen ja sähköpostitusominaisuuksien toteuttamista.
- Tuoteprojektin asiakassovelluksen raportointitoimintoja tulisi kehittää siten, että prosessiaktiviteetit voidaan viedä ohjelmallisesti MS Project-tiedostoformaattiin. Tämä toiminto nopeuttaisi prosessin käyttöönottoa.

Syksyllä 2007 Nokia Networks Siemens päätti uudesta prosessinkehitystyökalusta ja QPR Process Guide sovelluksen käytön lopettamisesta, joten Process Guide sovellukseen ei tulla tekemään vuoden 2008 uusia prosessipäivityksiä.

QPR järjestelmän alasajo ei kuitenkaan estä tämän järjestelmän käyttöä, vaan Prosessinhallintajärjestelmää käyttävät projektit voivat edelleen käyttää järjestelmää aina projektin päättymiseen saakka. Osa projekteista on useamman vuoden mittaisia, eli tämän järjestelmän alasajo tapahtuu aikaisintaan 2010. Uuden prosessinhallintatyökalun toiminnoista ja sen tietojärjestelmän sovittamisesta Prosessinhallintajärjestelmään tehdään tällä hetkellä kartoitusta ja kannattavuustutkimusta.

9 YHTEENVETO

Tässä insinöörityössä suunniteltiin ja toteutettiin toimitusvalmiusprosessin kehitysryhmälle prosessinhallintaohjelmisto, jonka tarkoituksena oli yksinkertaistaa prosessiin kuuluvan sähköisen materiaalin päivitystyötä, vähentää prosessipäivityksiin tarvittavaa työmäärää sekä parantaa sähköisen materiaalin käyttöä tuoteprojekteissa. Insinöörityö toteutettiin projektina, jossa ensin tutkittiin prosessinkehitykseen liittyvää yritystoimintaa ja eri teknologiavaihtoehtoja tuotettavan järjestelmän toteuttamiselle.

Projektissa toteutetun Prosessinhallintajärjestelmän tietokantaan ladattiin ensimmäinen virallinen toimitusvalmiusprosessi kesällä 2007 ja järjestelmä otettiin käyttöön kahdelle pilottiprojektille. Niiltä saadun palautteen perusteella sovellukseen tehtiin päivitys ja sovittiin käytön laajentamisesta kaikille alkaville tuoteprojekteille.

Prosessinhallintajärjestelmän toteuttaminen oli haastavaa ja mielenkiintoista, sillä järjestelmän toteutuksessa päästiin soveltamaan entuudestaan tunnettujen tekniikoiden parhaita puolia ja laajentamaan kokemusta MS Access- ja VBA-ohjelmoinnin saralla. Koko projektille alun perin varattu 6 kuukauden toteutusaika osoittautui liian lyhyeksi, sillä järjestelmään toteutettiin melko laajoja toimintoja, joiden toteuttaminen ja testaaminen veivät aikaa.

Onnistunut ohjelmistoprojektin läpivienti sekä uusien tekniikoiden oppiminen jättivät hyvän mielikuvan tietokantasovellusten toteutuksesta MS Accessilla ja Visual Basic for Applications-ohjelmointikielellä.

LÄHDELUETTELO

[1] Cardoza, Patricia; Hennig Teresa; Seach, Graham; Stein, Armen: *Access 2003 VBA Programmer's Reference*. Wiley Publishing Inc., Indianapolis 2004.

[2] Intrinsyc Software International Inc.: *J-Integra for COM Examples*. J-Integra. WWW-dokumentti. http://j-integra.intrinsyc.com/products /com/examples.asp. 2006. Luettu 22.11.2006.

[3] Health Market Science: *Jackess JavaDoc API documentation*. Jackess Project. WWW-dokumentti. http://jackcess.sourceforge.net /apidocs/index.html. 2006. Luettu 18.11.2006.

[4] Honkanen, Tatu: *Alkoholipankin käyttöliittymä*. Insinöörityö. Helsingin Ammattikorkeakoulu, Sähkö- ja tietoliikennetekniikka, Ohjelmistotekniikka. Helsinki. 2004.

[5] Koljonen, Seppo: *Interaktiivinen kalenteriohjelmisto www-ympäristöön*. Insinöörityö. Helsingin Ammattikorkeakoulu, Sähkö- ja tietoliikennetekniikka, Ohjelmistotekniikka. Helsinki. 2002.

[6] Liukkonen, Vesa: *Insinööritöiden seurantajärjestelmä*. Insinöörityö. Helsingin Ammattikorkeakoulu, Sähkö- ja tietoliikennetekniikka, Ohjelmistotekniikka. Helsinki. 2005.

[7] McFedries, Paul: *Visual Basic for Applications Unleashed*. SAMS Publishing, Indianapolis 1997.

[8] Microsoft Corporation: Visual Basic Developer Center. Microsoft Developer Nerwork (MSDN). WWW-dokumentti. http://msdn2.microsoft .com/en-us/vbasic/default.aspx. 2006. Luettu 30.11.2006.

[9] MySQL AB: *Embedded MySQL Case Studies*. MySQL.Com. WWW-dokumentti. http://www.mysql.com/customers/?embedded. 2007. Luettu 15.01.2007.

[10] MySQL AB: *MySQL Community Edition downloads*. MySQL.Com. WWW-dokumentti. http://dev.mysql.com/downloads/mysql/6.0.html. 2006. Luettu 15.11.2006.

[11] Nokia Oyj: *Nokian sisäinen Tuoteprosessin ja Toimitusvalmiusprosessin prosessikuvaukset, koulutusmateriaalit ja prosessinkehitysympäristön ohjeet*. Nokia Networks. Helsinki 2006. Luettu 1.9.2006. Luottamuksellinen.

[12] QPR Software Plc *QPR Process Guide Information pack and Brochure*. QPR.Com. PDF-dokumentti. http://www.qpr.com /Products/QPRProcessGuide/QPRProcessGuide_Features.html. 2006. Luettu 11.9.2006.

[13] Rajamäki, Juhani: *Tietokantaohjelmointi-kurssimoniste*. Moniste. Helsingin Ammattikorkeakoulu. Helsinki 2004.

[14] The Apache Software Foundation: *Apache POI - Case Studies*. The Apache POI Project. WWW-dokumentti. http://poi.apache.org/casestudies.html. 2006. Luettu 20.11.2006.

[15] Viescas, John L.: *Microsoft Access 2000 käyttäjän käsikirja*. Gummerus Kirjapaino Oy, Jyväskylä 2000.

[16] Viescas, John L.: *Microsoft Office Access 2003 Inside Out*. Microsoft Press, Redmond 2004.